An Efficient Framework of Utilizing the Latent Semantic Analysis in Text Extraction

Abstract The use of the Latent Semantic Analysis (LSA) in text mining demands large space and time requirements. This paper proposes a new text extraction method that sets a framework on how to employ the statistical semantic analysis in the text extraction in an efficient way. The method uses the centrality feature and omits the segments of the text that have a high verbatim, statistical, or semantic similarity with previously processed segments. The identification of similarity is based on a new multi-layer similarity method that computes the similarity in three statistical layers, it uses the Jaccard similarity and the Vector Space Model (VSM) in the first and second layers respectively, and uses the LSA in the third layer. The multi-layer similarity restricts the use of the third layer for the segments that the first and second layers failed to estimate their similarities. Rouge tool is used in the evaluation, but because Rouge does not consider the extract's size, we supplemented it with a new evaluation strategy based on the compression rate and the ratio of the sentences intersections between the automatic and the reference extracts. Our comparisons with classical LSA and traditional statistical extractions showed that we reduced the use of the LSA procedure by 52%, and we obtained 65% reduction on the original matrix dimensions, also, we obtained remarkable accuracy results. It is concluded that the employment of the centrality feature with the proposed multi-layer framework yields a significant solution in terms of efficiency and accuracy in the field of text extraction.

Keywords: Automatic Text Extraction, Multi-layer Similarity, Latent Semantic Analysis, Vector Space Model.

Abbreviations

ATE	Automatic Text Extraction
ATS	Automatic Text Summarization
CR	Compression Rate
LSA	Latent Semantic Analysis
MLS	Multi-Layer Similarity
NLP	Natural Langauge Processing
RSI	Ratio of Sentences Intersections
VSM	Vector Space Model

1 Introduction

In 2010, The Compare Business Product website[1] stated that the database of the World Data Centre for Climate stores 220 terabytes of text, and the Library of Congress contains 5,000,000 digital documents (20 terabytes of text) and every day the library records 10,000 items. In 2017, the World Wide Web website[2] published that the number of pages indexed by Google and Bing search engines reached 4.61 billion pages. Connecting these facts with what the Rayner et al. in [1] found, they studied the average reading speed for an adult, and they found that the average reading time for an adult is 200-400 words per minute, and they concluded that with the massive volume of text data available the reader should raise his/her reading speed to 4200 words per minute.

The reading that leads to a deep understanding of the text cannot reach the limit that Rayner et al. set and removing the insignificant parts of the text will be more useful. The volume of text data and the shortage of human ability to read a large

[1] Http://www.comparebusinessproducts.com/fyi/10-largest-databases-in-the-world (Seen in 8-8-2017)

[2] http://www.worldwidewebsize.com (Seen in 8-8-2017)

number of pages in short time necessitate the need to create automatic summarization systems that can extract the salient parts of the text and reduces the time necessary to capture the main ideas.

Automatic Text Summarization (ATS) is not a new field of research for computer scientists; it has been studied since the fifties of the previous century [2, 3]. But, the massive volume of text data and the need to obtain accurate and fast results push the researchers to pay more effort in this field and gain more precise and efficient results. From [4 - 8], we abstracted that the ATS is a computerized process of condensation that yields a shorter version of the original text and keeps the core meaning and the main ideas reserved (also see [9]). In this work, we concentrate on a particular kind of ATS called the Automatic Text Extraction (ATE). The ATE is the kind of summarization that copies the salient parts of the text without adding any information or changing the text structure (see [5]).

1.1 Background

Mani [5], Mei and Chen [7], and Gambhir and Gupta [10] classified the automatic summaries. They organized the summaries according to several factors such as the content, the structure, the purpose, and the type of input stream. The classifications can help us to understand the useful type of summary that is necessary for a particular area. The classifications are:

Abstract vs. Extract: The summary that produces by copying the salient sentences found in the document(s) is called Extract. The extract is a condensed version that is extracted from the original document and that may not be suitable for reading, but it is helpful for the other areas such as the Information Retrieval and the Natural language processing whose main concern is the contents not the coherence of the text. The abstract is the summary that specifies the salient sentences and rewrites and reorders the ideas to produce a summary resembles a human-produced summary. The abstract is a coherent text that is produced to reduce the time needed to read a book or a newspaper, and it requires a deep understanding of the text and does not depend only on copying some of the salient phrases or sentences [5, 11, 12].

Indicative vs. Informative: the Indicative summary is a kind of summary- mainly used in the search engine- that gives the user selective parts of the retrieved documents, and according to these parts the user may discard the document or read the full document and considers it as relevant. The Informative summary adds more details and tries to investigate all the salient information [10].

*Single vs. Multi-document***:** The third way of classifying the summaries is related to the input of the summarization process. If the input is a single document and the generated output is a single summary, then the process is called Single-Document summarization, and if the input contains more than one document and the purpose is to summarize them in one summary, then the process is called Multi-Document summarization [5].

Focused vs. Generic: the Focused summary relies on a specific factor (topic, article title, user query, or even user status). In this kind of summary, the summarization process will be directed to give more attention to any sentence or phrase in the original document that contains information about a predefined factor(s). On the contrary, the Generic summary is a miniature version of the original text without considering any initial requirements during the summarization process [5].

Statistical and Linguistic approaches have been used to build automatic summarizers. *The Statistical approaches* give the terms and the sentences quantitative scores based on the presence of certain features. The statistical approaches that are used in text summarization include: Vector Space Model [13 - 15], Machine Learning [16], Fuzzy Logic [17], Latent Semantic Analysis [17, 19, 20, 21, 22, 29, 45], and Neural Networks [23 - 25]. *The Linguistic approaches* are language dependent, and they extract the summary based on the linguistic features or structures (see [26 - 28]).

In this paper, we are interested in the statistical approaches especially the Latent Semantic Analysis. LSA can simulate the way people acquire knowledge and meaning through the correlation of facts from several sources [29]. LSA goes beyond the traditional statistical approaches and approximates the semantic meaning of the sentence through the exploration of the meaning of each word found in that sentence or document. The LSA assumes that the text meaning is the collection of the meanings of the words found in the text and those meanings should be captured by analyzing huge text corpus. In the literature, researchers showed that the LSA is a significant tool in text mining because it addresses the semantic meaning that solves some of the problems of the other statistical approaches [17, 19, 20, 21, 22, 29, 45]. As stated in [30] the LSA was proposed in the field of Information Retrieval and Natural Language Processing (NLP) to solve two main problems in the VSM model, the synonyms and polysemy:

(1) *Synonyms problem:* it arises when two or more different terms have the same meaning. For example, the Arabic words شُجاع, جَسُور, جَريء, and مقدام all of them have one meaning “brave.”

(2) *Polysemy problem:* it arises when one word has several meanings. For example, the Arabic word عين can convey several meanings, eye, allocated, appoint, spy, and spring of water.

The LSA computes the similarity between to texts by identifying the shared topics (concepts) between them. The LSA reduces the original terms-documents matrix to a terms-concepts(or documents-concepts) matrix in a lower semantic space. It uses the Singular Value Decomposition (SVD) to map the original terms-documents matrix to a reduced matrix that approximates the original one. The SVD is an algebraic matrix factorization technique for matrix reduction. The SVD decomposes a rectangular, huge, and sparse matrix and produces a smaller matrix with a small rank. Given a terms-documents matrix, the SVD decomposes it to the multiplication of three simple matrices, the first one represents the terms-concepts matrix, the second represents the documents-concepts matrix, and the final one represents the strength of the concepts with respect to each term or document [30].

The SVD is a powerful and effective reduction mechanism, but it has tangible drawbacks related to the huge space and time complexity requirements. Donga et al. [31] showed that 70% to 90% of the execution time of the modern applications that use the LSA goes to the running of the SVD. He et al. [32] detailed the time complexity analysis of the LSA Similarity. They found that the time complexity is the minimum of $\{t^2d, td^2\}$ where t is the number of terms in a huge corpus and d is the number of documents.

To solve the LSA time consumption problem, we propose to integrate the LSA with two statistical techniques, the Jaccard coefficient and the VSM models. The Jaccard Coefficient measures the ratio of shared terms between two text segments (documents), and the VSM gives each term a score (based on the term frequency in the document and the distribution of the term over the whole documents) and computes the similarity between the text segments based on these scores. Jaccard is simple, efficient, and suits the portions of the text that contains repetitive terms, but it gives all the words the same degree of importance and it cannot determine the important terms found in any sentence. The VSM model is the most widely used model in information retrieval and natural language processing [13,14,64,65,66], and it solves the problem found in the Jaccard coefficient by using a robust weighing scheme, but it faces some problems related to the semantic meaning of the terms such as the synonyms and polysemy. The Jaccard, VSM, and LSA techniques are separately tested and implemented in the literature of text summarization, but we collected them in an effective and efficient sentences similarity model. The model is called the Multi-Layer Similarity model (MLS). We showed in this research that the use of the Jaccard, VSM, and LSA in a separate manner either yields insignificant results or requires unreasonable time and space, and if we adapt them to work together, the performance and accuracy will increase.

The primary concern in the developed model is the extraction efficiency. Therefore, we chose to process the input text in layers. The choice of the similarity technique that should be used in each layer is based on the time complexity of the three similarity techniques (the Jaccard, VSM, and LSA). In our method, the propagation of the text extraction process starts from the simple and less time-consuming technique (Jaccard coefficient) to more advanced and time-consuming statistical techniques (VSM in the second layer and the LSA in the third layer). The Jaccard will process all the sentences but this will not hurt the overall efficiency because it requires O(n*m) where n is the number of sentences in a given document and m is the number of terms in each sentence. The second layer needs more computational steps because it first computes the weights of the terms over the whole corpus, then it computes the cosine similarity between the sentences found in each document. However, The VSM model will only process the sentences that are not deleted in the first layer, which participates in accelerating the VSM process. The third layer, which is the most time-consuming model (LSA) [31, 32], will process the sentences that are not deleted in the first and second layers (only 35% of the text as we showed in section 6.4).

The choice of three layers (not more than three) is based on the thought that inserting more layers adds more time penalty. The three layers are considered sufficient because through these levels of statistical analysis the verbatim existence of the terms is recognized (Jaccard processing) and the important terms are identified (VSM processing) and the semantic meanings of the terms are considered (LSA processing).

The paper also addresses an important issue related to text summarization (or even the text mining in NLP), which is the assessment of the output quality. As stated by Jing et al. [33], we cannot assume the existence of a typical or gold answer during the evaluation process. Sparck and Galliers in [34] and Mani in [5] stated that the evaluation strategy for any summarization systems should include means to measure:

(1) The summary length (Condensation Rate or Compression Ratio (CR)): This equals the summary length divided by the full-text length.
(2) The salient parts: this measures if the automatic summary preserves and maintains the main ideas.

As mentioned by Sparck and Galliers [34], the two main approaches that are experienced to evaluate the quality of the automatic summaries are Intrinsic and Extrinsic approaches. *Intrinsic approach* was used in [4, 6, 13, 26, 35, 36, 37]. It uses the human-generated summary as an ideal answer and compares the system-generated summary against the human-generated summary to find the resemblance between them. The *Extrinsic approach* [38, 39] assesses the impact of the automatic summary on the other NLP fields such as the Topic Detection, Question Answering systems, and Information Retrieval.

The intrinsic manual approach is widely used in the past, but this type of evaluation is affected by many factors such as the evaluator background or education level [13], and the evaluator opinion or point of view [40]. In addition, it is expensive and takes a lot of time. Therefore, the researchers implemented the intrinsic approach in many evaluation tools: Rouge [41], SEE [42], and MeadLeval [43].

Rouge Evolution Toolkit is an evaluation software developed by Lin [41]. It stands for Recall-Oriented Understudy for Gisting Evaluation, and it evaluates the quality of the automatically generated summaries by comparing them with human-generated summaries (called reference, gold, or ideal summary). Rouge counts the number of intersections between the computer-generated summary and the gold summary created by humans (or by another system for comparison purposes). Rouge statistically measures the resemblance, but it cannot indicate the percent of complete sentences from the gold summary that appears in the automatic summary. The produced recall and precision measurement values increase by the existence of any sequences of n-grams, words, or phrases.

Rouge 2.0 adapted the recall and precision to be applicable for the text extraction evaluation tasks. The precision measures the accuracy of the outputted answer set relative to the contents of the answer set itself, whereas the recall measures the accuracy of the answer set relative to the contents of an optimal or gold answer. In Rouge, the outputted answer set is called the system summary, and the optimal or gold answer set is called the reference summary. According to Lin in [41], Rouge computes the recall and precision using the following equations:

$$Recall = \frac{number\ of\ overlapping\ unigrams}{total\ number\ of\ unigrams\ in\ reference\ summary}$$

$$Precision = \frac{number\ of\ overlapping\ unigrams}{total\ number\ of\ unigrams\ in\ system\ \ summary}$$

$$F - score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

In the literature of ATS, the Rouge is widely used to evaluate the summarization systems, and from our list of references, the Rouge was used in [16, 19, 21, 22, 46, 61].

SEE was developed by Lin [42], and it stands for Summary Evaluation Environment. It is a software package that facilitates the evaluation of the computer-generated summaries. It provides an interface with two panels one shows the computer-generated summary (called peer summary) and the second shows the reference summary (called model summary). Assessors evaluate each sentence in the peer summary panel and then compare it with the model summary. Each sentence in the peer summary takes one of five values (All, Most, Some, Hardly, and None) depends on the degree of similarity to the model summary sentences. The assessors evaluate the summary contents, grammar, coherence, and cohesion. The tool facilitates the manual evaluation**.**

Another tool used to assess the quality of the computer-generated summaries is the MeadLeval. The MeadLeval toolkit employs a data structure called the “extract file”, this file stores important information about the extracted sentences. Similar to Rouge and SEE, MeadLeval compares the computer-generated summary with the ideal summary. MeadLeval supports many evaluation metrics such as Recall, Precision, Kappa, and others.

In Rouge, SEE, and MeadLeval, the recall focuses on the portion of relevant retrieved text relative to the reference summary, and the precision focuses on the portion of relevant retrieved text relative to the automatic summary. As stated by Ramanujam and Kaliappan [44], both recall and precision do not take the size of the automatic extract in consideration during the evaluation process. Theoretically, the recall increases if the size of the automatically generated extract gets close to the size of the original document. In our approach of extraction, we create extracts with variable sized length, and in some cases, the CR was high (75% and above). Therefore, it's impractical to consider the recall and precision as the only measures of relevance. Furthermore, the employment of any sequences of n-grams or words (Not complete sentences) in the computation of the recall and precision yields inaccurate results. In this paper, we propose a new evaluation technique called the Containment evaluation that measures the percent of complete sentences shared between the automatic and the reference summary and considers the size of the automatic summary. The Rouge version that seems to be close to our new evaluation is Rouge-L. But, this version does not ensure the consecutive order of the terms, and it measures the longest common subsequence (not necessarily complete sentences) between the gold and the automatic summary.

Finally, according to the categorization of the summaries stated in [5, 7, 10], we classify the automatically generated summaries by our method as:

(1) Extract: because we copy certain parts from the original document, and we do not restructure the sentences or change their order.
(2) Informative: because we try to extract all the salient parts of the text, not some of them.
(3) Free size text: because we work on the sentence level, and any text contains two sentences or more can be processed by our method of extraction.
(4) Generic: because the extraction process does not focus on a certain factor.

1.2 Main Contributions

In this paper, we aim to build an ATE method that produces a variable-sized summary and keeps all the main sentences reserved by efficient use of the LSA. The main contributions of this research are summarized in the following points:

***Build up a framework of the text analysis*:** We established a framework on how to use the statistical approaches in text mining in such a way that increases the performance and decreases the complexity. The statistical approaches range from a simple measuring of terms overlaps to a complex semantic meaning analysis. We showed that the use of terms overlaps and the traditional statistics based on tf-idf analysis in an individual manner (without semantic investigation) failed to remove a reasonable portion of the text (CR = 68%, only 32% from the text is removed). And in [17, 19, 20, 21, 22, 29, 45], the authors showed that the semantic analysis of the text using a powerful semantic analyzer such the LSA increases the accuracy but the time complexity was the challenge. Therefore, the proposed MLS method draws a map on which kind of text analysis should be used for each part of the text. The rate of verbatim existence (Jaccard) and the statistical analysis (VSM) can be used in certain parts of the text, but they are not enough, we have to supplement them with a semantic analyzer that solves the complicated cases. Our claim was to process the text in layers. The lower layer depends only on the rate of shared terms, the second layer used traditional statistical analysis (tf.idf weighting and cosine similarity), and the upper layer works on the remaining parts of the text and explores the semantic meaning of the text. In the MLS method, the time-consuming procedure, which is the LSA, is used in minimum, and this increases the acceptability of employing the LSA in the Text Mining fields. Our results prove our claim, the results in Fig. 2-10 (section 6) were comparable with the results of the existing extraction system that used the classical LSA semantic analyzer, knowing that our method used the LSA for only 35% of the original text. This implies that we obtained comparable accuracy in an efficient way (Fig. 12 and 13 and Table 12 in section 6).

***Developed algorithm for analyzing the centrality feature of the sentence*:** The sentence centrality feature that is used in this research appeared in [18, 24, 46]. The sentence centrality measures the importance of the sentence with respect to the other parts of the text. In [18, 24, 46], the authors either used simple vocabulary overlaps to determine the sentence centrality, which did not give us the actual benefit of this important feature, or use it as a feature from a group of features, which makes their effect unclear, for example, both [24, 46] used the centrality features with more than ten features (see Table 1). Therefore, the precision and recall values appeared in [18, 24, 46] are produced from a combination of features not individually from the centrality feature. Thus, we consider our work as the only work that uses the centrality as the only distinguishing feature and the only work that uses efficient semantic analysis to measure the sentence centrality. The centrality is determined by combining the vocabulary overlaps with the VSM and the LSA in a multilayer similarity scheme. The centrality feature is the only condition that controls the deletion of similar sentences in our deletion process (section 3.2.3), and the results of using the centrality feature are promising (at CR=42% the Avg_RSI = 55% the ratio of complete sentences, Avg_Recall = 48%, Avg_Precision =40%).

Robust evaluation strategy: for fixed sized extracts or summaries, the Rouge tool gives a significant indication of the extraction accuracy, but with variable sized automatic extracts the Rouge gives misleading assessment because it measures extracts of different sizes. In this paper, the Rouge assessment of the Jaccard based extraction and the VSM based extraction gave results higher than the LSA analysis (Jaccard recall=79%, VSM recall = 63%, and LSA recall = 60%), which seems illogical and incompatible with the vast majority of research in this field. The Rouge evaluation cannot judge accurately the percent of shared sentences between the automatic and the reference extracts, and it does not consider the size of the extract. In this paper, we propose the Containment evaluation that measures the percent of the complete sentences that are shared between the automatic and the reference summaries and takes into account the size of the automatic summary. Using the Containment evaluation, we found that 97% of the Jaccard extracts sentences are found in the manual extracts but at CR=79%, and we found that our method of extraction achieved 65% Containment (65% of the manual

extracts sentences appeared in the automatic extracts) at CR=42%. Note that our evaluation gave a more fair judgment.

The remaining sections of this paper will be as following: section 2 discusses the related work. Section 3 explains in details the contributions and describes our method of text extraction. Section 4 details the experiment used to test our method of extraction such as the datasets and the experimental procedure. Section 5 presents the results that are generated in the experiment section. And in section 6, we evaluate the results using several evaluation schemes. Section 7 discusses the main findings of the evaluation section. Finally, section 8 summarizes the main conclusions and the future work of this paper.

2 Related Work

Luhn in [3] investigated the ATS since the mid of the previous century. Statistical and linguistic features of the text have been examined to determine the salient sentences. Statistical approaches are widely used, they give each sentence a numerical score based on the existence of a certain feature or a combination of features and rank the sentences in the document based on the computed score. The features include the term frequency [3], terms position [47], cue phrases [4, 35], signature words [48], phrase frequency [2, 3, 49], lexical cohesion [50], discourse structures [18, 51], the presence of certain word types [52], the sentence centrality [18, 24, 46], and others. Table 1 shows the features examined in the literature of ATS. From the surveyed publications we found no consensus about the ideal combination of features and heuristics that gives the highest score in the weighting stage. Ferreira et al. [46] stated that the best features combination includes: the word frequency, the inverse term frequency, the lexical similarity, and the sentence length. Whereas, Meena and Gopalani [9] found that the sentence location, the named entities, and the proper noun are the most effective features in identifying the salient elements in the text. Lin in [53] tested a system called SUMMARSIT to generate summaries of multilingual input texts. This system combines existing natural language processing methods with symbolic world knowledge and information retrieval technique to automatically learn the significant combination function.

The vast majority of the research reported earlier relies on statistical approaches, Yanmin et al. [26] followed another direction toward the linguistic analysis. They investigated the cohesion structure of the text by locating the lexical chains that were obtained from HowNet and TongCiCiLin lexical databases. Tayal et al. [54] used POS tagger and NLP parser to analyze the sentence before finding its semantic meaning using the WordNet. Alruily et al. [28] utilized a linguistic feature of the Arabic language to delete all the text located between the verb and its object, which takes the form of a preposition phrase. Shams et al. in [55] merged the statistical and linguistic methods in one summarization system. They employed Stanford POS Tagger and a term co-occurrence graph to find the subject of the sentence. However, the linguistic approaches are language dependent, and almost we cannot generalize and use them for another language.

El-Haj and Hammo in [13] utilized the Information Retrieval techniques to extract the important portions of the document. They experienced the use of the VSM and the cosine similarities to summarize the selected document. The initial step starts by selecting the document that matches a given query and then a score is computed for each sentence based on the similarity between the query and the sentence in that document. In more details, they represented the query in vector structure and did the same procedure for each sentence and apply the cosine similarity between them. Indeed, the use of the user queries may cause a problem to the extraction system especially when the topic being searched is mysterious or new to the user. This kind of extraction is helpful for the indicative summaries but not for the informative summaries. Ghwanmeh et al. [14] replaced the user query with the first sentence in the document and applied cosine similarity to find the sentences in the document that resembles the first sentence. Ghwanmeh et al. [14] assumed that the first sentence is the topic sentence, but normally it is the introductory sentence that introduces the topic sentence. Sometimes, it represents the hook sentence -especially in the news and stories - that attracts the user attention[3].

Bushy path and aggregate similarity are Information Retrieval-related techniques that were used by Ferreira et al. [46]. In the Bushy path method, each sentence is represented as a node on a map, two nodes (sentences) in the map are connected if the similarity between them is greater than zero, and the label above each link represents the similarity value. The bushy path method counts the number of the links going out from each node and the node with the greatest number of links takes the highest score. Aggregate similarity used the same map, but it sums up the similarity values placed on the

[3] See http://laflemm.com/reso/introSentences.html (Last update of this page: Feb. 27, 2014)

links. Both the bushy path and the aggregate similarity maximize the value of small similarities and equalize them with the large similarity values. For example, the bushy path gives more value to the node that has two links than the node that has one link even if the similarity in the latter case reaches 100%, and aggregate similarity gives the node that has two links with aggregate similarities 70% more value than the node that has one link with 69% similarity.

Other statistical techniques used to extract semantically related sentences in a document are the LSA [17, 19, 20, 21, 29, 45] and the Neural Network [12, 23, 24, 25, 56]. Different Neural Network (NN) methods have been used to extract summaries from the text document. These methods include: the deep auto-encoder NN method [25], the sequence to sequence model [12], the recurrent NN model, the Feed Forward NN model, the Gaussian mixture model network [24], and the probabilistic neural network [23].

The LSA, which is the core of this paper, is a mapping process in which we map terms to their topics. Mashechkin et al. [19] used the LSA to generate generic extracts, and they integrated the LSA with non-negative matrix factorization to preserve the internal structure of the text. Yang et al. [20] used LSA to reduce the effect of the synonyms and polysemy problems that are experienced in the VSM model. Wang and Ma [21] added more semantic information to obtain accurate sentence selection when they chose the sentences that best describe the concept and contain certain terms that best represent it. Ba-Alwi et al. [22] experimented the LSA for the Arabic language, and they achieved 46% average Rouge. Babar and Patil [17] compared the LSA with the Fuzzy logic in scoring and selecting the summary sentences, and they found that the accuracy of the LSA was the highest, Ngoc and Tran in [29] integrated the LSA with Dennis coefficient to semantically classify the English text.

In [17, 19, 20, 21, 22, 29, 45], the time complexity of running the LSA procedure was not addressed, which represents the main challenge of employing the LSA in any NLP application. Recently in 2017, Gambhir and Gupta [10] reviewed almost all the automatic text extraction techniques proposed in the literature. They listed the published papers with their approaches and results. The survey showed that a great effort has been done, but the research that addressed the semantic analysis did not consider the high time complexity of applying the LSA in the text extraction. Therefore, the surveyed publications represent a good starting point because they proved that the LSA is a powerful mechanism to extract a summary from the text. We continue from this point, and we built an extraction system (called MLSExtractor) that efficiently uses the LSA. The MLS is a reduction step applied to the original terms-sentences matrix and produces a lower dimensional matrix. The MLS precedes the SVD execution, so its effect is reflected directly on the execution time of the SVD.

3 New Method for Extraction

In this section, we describe the MLS model of automatic text extraction. The MLS statistically measures the verbatim, statistical, or semantic resemblance between any two sentences or paragraphs and deletes the repetitive sentences. It is a self-extraction model that extracts the main sentences without the influence of the linguistic features, the text structure, and the user intervention. Therefore, it is a language, domain, and user independent extraction method. In the MLS, we do not use a reference sentence as a base for the extraction (such as the user query [13] , the first sentence [14], or the title of the document [18, 23, 35, 53]), and the system will not oblige to take a certain direction during the extraction process. Thus, the output will include a variety of information depending on what the document contains.

Another important issue is the CR, which is normally fixed [18, 24, 27, 37, 51, 58] or user predetermined [43, 57]. In MLS extraction, the output is a variable-sized extract that contains the main ideas found in the documents. The fixed CR forces the system to return a certain number of sentences or a predetermined ratio of the text and this may cause the systems to neglect certain salient sentences because the summary length exceeded the CR. Our claim states that the CR should depend on the richness of information found in the document and our model implements this idea.

3.1 Basic Concepts

Important definitions for computing the term weights and the similarities between the pairs of sentences are introduced. Then, the definition and lemmas necessary to select the extract's sentences are presented.

In this research, the developed method of extraction measures the similarity in four levels of complexity, the rate of verbatim existence (Jaccard coefficient), traditional statistical (VSM model), statistical with semantic analysis (LSA model), and multi-layer of statistical and semantic analysis (MLS model).

Table 1 the Extraction Features, Techniques, and Accuracy in our References

Ref	Technique	Features	Accuracy (Recall(R), Precision(P), F-score, Compression Rate(CR)
[3]	Statistical	Term frequency	Not mentioned
[13]	Statistical Method	Term frequency/inverse term frequency	Not mentioned
[15]	Statistical- Machine learning approach using Fuzzy and Vector method	Mean-TF-ISF / Sentence-to-Sentence cohesion / Sentence to centroid cohesion	Vector Method: At CR=10% R=21%, P=21% Fuzzy Method: At CR=10% R=28.2% P=29.6%
[16]	Statistical with Machine learning methods	Position / N-grams frequencies / query term / Wikipedia entity(titles of Wikipedia pages)	CR = three sentence Rouge-1 score = 0.52
[17]	Statistical Methods with the employment of Fuzzy logic and LSA.	Title words / Sentence length / Sentence position / Numerical data / Thematic words / Sentence to sentence similarity / Term weight / Proper nouns	CR did not computed Using fuzzy scoring: R=41%,P=86% Using LSA:R=44%,P=90%
[18]	Statistical (MCBA + GA) Statistical (LSA + T.R.M)	Word position / Positive keyword / Negative keyword / Centrality / Resemblance to the title	At CR 30% f-Score= 52% for CBA+GA, f-Score=40% for LSA+TRM
[23]	Statistical with Neural Network	Sentence Position / Keywords /negative keywords / Centrality /Similarity to the title /Proper noun / Numerical data /Sentence length /Pushy path /aggregate similarity.	At CR=10%, R=45% At CR=20%, R=46% At CR=30%., R=47%
[24]	Statistical	position / positive keyword / negative keyword / centrality / sentence resemblance to the title / aggregated similarity / Inclusion of name entity, sentence / inclusion of numerical data / sentence relative length / Bushy path of the sentence.	Using DUC 2001 dataset (*The maximum precision was obtained using GMM)* P(GMM) = 0.5902 CR=10% P(GMM) = 0.5936 CR=20% P(GMM) = 0.6046 CR=30%
[25]	Statistical, with the employment of NN	Term frequency	average Rouge 46%
[26]	Linguistic analysis	Locating the lexical chains obtained from HowNet and TongCiCiLin lexical databases.	At CR=10%, P=0.73, R =0.77 At CR=20%,P=0.71, R=0.74
[27]	Statistical and Linguistic	Normalized Phrase Words / Phrase Words / Phrase Relative Frequency / Word Relative Frequency. / Sentence Location / Phrase Location / Phrase Length / Contain Verb / Is It Question	At CR=25% R=52% P=71%
[28]	linguistic methods	Transitive verbs by prepositions.	Not mentioned
[35]	Statistical	Cue-word/ Sentence location / Title and heading words	Not mentioned
[37]	Statistical, with the genetic algorithms	Term frequency, Sentence position, Sentences length, similarity to the title	at CR=40% Avg Recall=55%, Avg Precision=45%, F_measure=54%.
[46]	Statistical	TF/IDF / Upper case /Proper noun / Word co-occurrence / Lexical similarity Cue-phrase /Inclusion of numerical data / Sentence position /Sentence centrality / Resemblance to the title / Aggregate similarity/Bushy path	Average R = 73% Average P = 40% Average F- Measure =(73%
[47]	Statistical	Word position	Not mentioned
[53]	Statistical	tf / tf.idf / Title and position / IR signature / Average lexical connectivity / Numerical data / Proper name, pronoun and adjective. / Weekday and month	The maximum F_ Measure value was 58% at CR 20%
[54]	Linguistic and Statistical Methods	Word tag(Subject, Verb, and Object) / Title or theme of the document / N-gram co-occurrence	CR Not mentioned F-scores = 14% Recall= 40%
[55]	Merged statistical and linguistic methods	Statistical parameters: TF / Sentence weight / Subject weight. Linguistic methods: Employed Stanford POS Tagger and a term co-occurrence graph to find the subject of the sentence.	At CR= 30% R=65%
[56]	Statistical, with the employment recurrent NN	Tem frequency in the sentence / The sentence length	CR not computed Rouge 0.1, R=40% Rouge 2.0, R=26%
[57]	Statistical- Rhetorical Structure Theory	word frequency / sentence location / title keyword	At CR=31% P=66% R=70% F- Measure =67%

Jaccard Coefficient Similarity: we used the Jaccard coefficient as the first level of similarity computation because it can process the parts of the text that contain a sufficient number of shared terms. We want to reduce the input of the upper levels, the VSM in the second level, and the LSA in the third level. The Jaccard similarity uses a simple statistical equation to measure the ratio of shared terms between two sentences.

Definition 1. Given a text document T as a set of sentences, $T = \{S_1, S_2, S_3, \dots, S_M\}$, and $S_i, S_j \in T$ are two sets of words (terms) such that $S_i = \{t_1, t_2, t_3, \dots, t_x\}$ and $S_j = \{t_1, t_2, t_3, \dots, t_y\}$, if $S_i \cap S_J \neq \emptyset$. Then, the Jaccard similarity is defined by the following equation:

$$\text{Jac}(Si, Sj) = \frac{|Si \cap Sj|}{|Si \cup Sj|} \qquad (1)$$

Example 1 form doc9: Given S_2 = {زوج, رحل, عصي , رحب, خوي, عرف , صور, رحب, خوي , ثور , علم, موسيقى, غني, عرب,شكل } and S_{11} = {فلم, يبرز, ألف, لحن, جمع, عدد, جرب, خوض, عام, عصي, زوج, وفي, يبعد, غني, كبر, جمع, قدم, رحب, زود, أبن, روس, شكل, عمل, صوف, زكي , وهب, علم, موسيقى , عرب , موسيقى, سقي, خوص, موسيقى, نمط, خلق, قدر, وهب, يبرز, وما, زول , سور, فنن, سمر, نجح}

$$\text{Jac}(S_2, S_{11}) = \frac{8}{52} = 0.15$$

Note that the two sentences have different lengths. Thus, for length normalization, we changed equation 1 to be:

$$\text{Jac}(S_i, S_j) = \frac{|\text{Si} \cap \text{Sj}|}{\min_{S_i, S_j \neq \emptyset}(|S_i|, |S_j|)} \qquad (2)$$

Back to example 1.

$$\text{Jac}(S_2, S_{11}) = \left(\frac{8}{15}\right) = 0.54$$

This equivalent to saying that 54% of the terms of S_2 are found in S_{11}.

The Jaccard coefficient is a statistical scheme that can be used to measures the similarity between two sets of terms (words). But, it does not consider the importance of the term with respect to the other terms or the whole document. It sees the sentence as a set or bag of words without taking into account the terms meaning, orders, or relationships. However, we employed the Jaccard coefficient because for certain cases the terms overlaps can give a significant indication about the similarity if the overlap was large.

VSM Similarity: The second level of similarity computation uses the VSM. The VSM estimates the weight of the terms based on their frequency in the text segment and their distribution over the whole segments found in the document(s). The parts of the text targeted by the second level of our analysis are the segments that contain terms that appeared frequently in those segments and distributed over a few numbers of segments. The second level processes more important terms than the terms processed in the first level (Jaccard) because the words that appear repeatedly and in all parts of the text are unlikely to be the subject of the text (think about the stop words). For example, if the document was talking about the term "networking", then the segments in the text that contain this term should have higher weight then the segments that do not contain it. The similarity will be reasonable between the heavy weighted segments that represent the main ideas of the document(s).

We used the VSM because it has been used deeply in the field of information retrieval (and gives remarkable results). The VSM used in information retrieval to measure the similarity between the user query and the documents being searched. In our employment of the VSM, we represented all the sentences in the document as vectors in the VSM model by giving their stems numerical weights. The term frequency (tf) and the inverse term frequency (idf) should be computed for every term, and these parameters determine the term weight.

Definition 2. Given a text document T as a set of sentences and $T = \{S_1, S_2, S_3, \dots, S_M\}$, and given a sentence S_i that has V_{Si} vector, $V_{Si} = (x_1, x_2, x_3, \dots, x_n)$, and x_i is the term weight of the i-th term in S_i , and given a sentence S_j that has V_{Sj} vector, $V_{Sj} = (y_1, y_2, y_3, \dots, y_m)$, and y_j is the term weight of the j-th term in S_j, and $S_i, S_j \in T$ then, the VSM similarity can be defined by the cosine of the angle between the vectors V_{Si} *and* V_{Sj}:

$$sim(Si, Sj) = \cos(V_{Si}, V_{Sj}) = \frac{V_{Si} . V_{Sj}}{|V_{Si}| . |V_{Sj}|} \qquad (3)$$

If we represent $\overline{s_i} = (w_{1,i}, w_{2,i}, w_{3,i}, \dots, w_{t,i}), and\ \overline{s_j} = (w_{1,j}, w_{2,j}, w_{3,j}, \dots, w_{t,j})$, where $w_{1,i}$ the weight of term 1 in S_i, $w_{1,j}$ is the weight of term 1 in S_j, and t is the number of the terms in the text T, then, the cosine similarity can be rewritten as follow:

$$\cos(\overline{s_i}, \overline{s_j}) = \frac{\overline{s_i} . \overline{s_j}}{|\overline{s_i}| . |\overline{s_j}|}$$

$$\cos(\overline{s_i}, \overline{s_j}) = \frac{\sum_{n=1}^{t} w_{n,i}\, w_{n,j}}{\sqrt{\sum_{n=1}^{t}(w_{n,i})^2}\sqrt{\sum_{n=1}^{t}(w_{n,j})^2}} \quad (4)$$

$$\text{Where } 0 \le cos(\overline{s_i}, \overline{s_j}) \le 1$$

The tf.idf is the most common scheme used to compute the vector weights. The term frequency (tf) and the inverse term frequency (idf) should be computed for every term in the corpus. Certain weighting scheme should be hired, and we used the weighting scheme proposed by Salton [30].

Definition 3. For any sentence $S_i \in T$, the $tf_{t,si}$ is the number of times the term t appeared in the sentence S_i.
The log of 10 normally normalizes the tf because the importance of the term does not increase proportionally with the tf, and the most common formula used to compute the tf is:

$$tf_{t,Si} = \begin{cases} 1 + \log_{10} tf_{t,si}\,, if\ tf_{t,si} > 0 \\ 0 \qquad\qquad otherwise \end{cases} \quad (5)$$

Definition 4. Given a term t, the idf_t is the number of sentences in T that contain t. and the idf_t is given by the following equation:

$$idf_t = \log\left(\frac{N}{df_t}\right) \quad (6)$$

Where N is the number of sentences in T

Definition 5. Given the $tf_{t,Si}$ and idf_t , the tf.idf weights $w_{t,si}$ of the term t is given by equation 7:

$$w_{t,si} = idf_t \,.\, tf_{t,Si} = \log\left(\frac{N}{df_t}\right).(1 + \log_{10} tf_{t,si}\ ) \quad (7)$$

Each sentence in the document is represented as a vector in the VSM model. After generating the whole vectors, we examine the angle between any two vectors. If the angle is large, this means that the two vectors are dissimilar and this appears as a small value of the cosine of the angle between them. On the contrary, if we find a high cosine value, this gives a significant indication about the resemblance between the two sentences.

The VSM model uses the statistical analysis approach and examines the text statistically, but it does not solve some problems related to the semantic meaning of the text (polysemy and synonyms). The VSM processes the synonymous words as different words, not as semantically related words. Therefore, we provide our deletion algorithm with the LSA similarity analysis that goes beyond the literal existence of the words.

LSA Similarity: In the third level of similarity computation, we used the LSA (as shown in Fig. 1). LSA starts by representing the text as a matrix in which the rows are the words, and the columns are the sentences[4] that contain those words. The cells entries reflect the importance of a certain word in a certain sentence. Then, the LSA applied the algebraic method SVD to make the necessary factorization analysis to reduce the number of rows and columns. The SVD decomposes the original matrix to three matrices as shown in Equation 8:

$$\boldsymbol{X} = \cup\, \boldsymbol{\Sigma V^T} \quad (8)$$

Where X is $i\ x\ j$ - the original matrix with rank k, k = min (i, j) , U is $i\ \ x\ i$ matrix that represents the left singular vector, Σ is a diagonal matrix, V is j x j matrix that represents the right singular vector. And in U, V^T the columns are orthonormal.

Definition 6. Given X as a set in a vector space and $X = \{\overline{v}_1, \overline{v}_2, \overline{v}_3, \dots, \overline{v}_n\}$, if $\forall \overline{v}_i, \overline{v}_j \in X, i \neq j, \overline{v}_i.\overline{v}_j = 0$, and $\forall v \in X$: $\|\overline{v}\| = 1$, then the vectors are called orthonormal.

In decomposing the matrix X, we transfer X from high dimensional space of rank k (terms-sentences space) to lower dimensional space of rank r (terms-topic space represented in U and sentences-topic space represented in V), $r < k$. The ∑ diagonal entries represent the singular values $\boldsymbol{\sigma}$ (the singular value is the square root of the eigenvalues λ) of X and they are sorted from largest in $\sum_{1,1}$ to smallest in $\sum_{i,j}$.

Definition 7. Let A be a $n\ x\ n$ matrix, λ is called the eigenvalue of A if there is a nonzero vector $\overline{x}$ such that $A\overline{x} = \lambda\overline{x}$, $\overline{x}$ is called the eigenvector of A corresponding to λ.
Note that the definition of eigenvalues and eigenvector required a $n\ x\ n$ matrix and X is $i\ x\ j$, so we want to obtain a square matrix from X to obtain the eigenvector decomposition.

Lemma 1. Let X be a $i\ x\ j$ matrix, then the matrix $X.X^T$ is square and symmetric.
Proof.

1. The dimension of X is $i\ x\ j$ and the dimension of $X^T$is $j\ x\ i$, then the dimension of $X.X^T$ will be $i\ x\ i$, this implies that $X.X^T$is a square matrix.

[4] We concern in this research paper by the sentence as the unit of text and perform all the similarity assumptions and calculations based on the sentences found in the document, because our deletion process works also at the sentences level to generate the required extract.

2. The symmetric means that the transpose of $X.X^T$ gives the same matrix.

$$(XX^T)^T = X^{T^T}X^T = XX^T \quad \blacksquare$$

According to the Definition7 and Lemma 1, we can make eigenvector decomposition, the vectors (columns) in U are eigenvectors of XX^T, and the vectors (columns) in V are the eigenvectors of X^TX (note that the eigenvalues of XX^T *and* X^TX are the same), so to find the factorization matrices mentioned in equation 8, we follow the following steps:

1. *Collect the eigenvalues(λ) of* XX^T *and the corresponding eigen vectors, normalize the vectors and store them as columns in U (construction of U)*
2. *Find the square root of λ's and store them in descending order in the diagonal of* $\sum = (\boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_k})$ *(Construction of* $\sum$*)*
3. *Collect the eigenvectors of* X^TX*, normalize the vectors and store them as columns in V (construction of V)*

The diagonal matrix $\sum$ reflects the strength of the concepts and it represents the core of the space reduction that the LSA performs. The main diagonal of $\sum$ contains the singular values, thus equation 8 can be rewritten as:

$$\boldsymbol{X} = \cup\, \boldsymbol{diag}(\boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_k})\boldsymbol{V^T}$$

The reduced SVD performs the required reduced rank approximation and transforms the matrix X_k with k rank to X_r approximation matrix with $r < k$ by setting the lowest $k - r$ eigenvalues in $\sum$ to zeroes. Note that the number of concepts equals to the number of $\boldsymbol{\sigma's}$ **(singular values)** and if some $\boldsymbol{\sigma's}$ values are very small this means that some of the concepts are not the core of the text (sentence or document), and the LSA will not consider them.

$$\boldsymbol{X_r} = \cup\, \boldsymbol{diag}(\underbrace{\boldsymbol{\sigma_1}, \dots, \boldsymbol{\sigma_r}}_{\text{r}}, \underbrace{\mathbf{0}, \mathbf{0}, \dots \mathbf{0}}_{\text{k-r}})\boldsymbol{V^T}$$

$$\xrightarrow{\boldsymbol{yields}} \boldsymbol{X_r} = U_r.S_r.V_r^T \qquad (9)$$

Where $\boldsymbol{X_r}$ *is i x j matrix and represents a reduced rank approximation of the matrix X,* U_r *represents the first r columns of U,* S_r *represents the upper r x r of* $\sum$*, and* V_r^T *represents the first r columns of* $\boldsymbol{V^T}$

After constructing the $\boldsymbol{X_r}$ matrix, the computation of the similarity between any two sentences $\boldsymbol{S_i}, \boldsymbol{S_j}$ can be accomplished by computing the dot product of the corresponding columns in $S_r.V_r^T$ matrix. See Lemma 2.

Lemma 2. Given T as a set of sentences $T = \{S_1, S_2, S_3, \dots, S_M\}$, X is a $i\ x\ j$ term-sentences matrix representing T, and $\boldsymbol{X_r}$ is the reduced SVD of the matrix X. The inner product of the columns vectors $\overline{S_i}, \overline{S_j} \in X, i \neq j$ is the inner product of the corresponding columns in $S_r.V_r^T$

Proof. Let $S_r = S_{rxr}$, $U_r = U_{ixr}$, and $V_r = V_{jxr}$, where i is the number of terms in X, and j is the number of sentences in T, r is the rank of X_r.

$$X_r^TX_r = (U_rS_rV_r^T)^T(U_rS_rV_r^T) = V_rS_rU_r^TU_rS_rV_r^T = V_rS_r^2V_r^T = (S_rV_r^T)^T(S_rV_r^T)$$

Note that S_r contains zeroes except the diagonal entries (diagonal matrix) $S_r = S_r^T$, and $U_r.U_r^T = I$, U_r, U_r^T are orthonormal. $\blacksquare$

After the dimension reduction, the inner product determines the similarity between the vectors of S (using $S_r.V_r^T$ matrix) or between terms (using $S_r.U_r$ matrix).

MLS Similarity: The final level of similarity computation is the Multi-Layer Similarity. If we consider the previously discussed similarity approaches, we can note that each one has significant strengths and weaknesses. Therefore, we proposed the MLS similarity calculation method that can benefit from the strong points found in each similarity approach discussed previously and employs the LSA in an efficient way that reduces the number of runs of the LSA extraction.

Normally, the LSA reduces the dimensions of the original matrix to a smaller number (100-500 or user-defined), this reduction –for example- from 1,000,000 to 100 or 500 may cause the loss of important sub-meanings especially for very huge and heterogeneous corpus. But, this problem will not make that much of effect in a single-document summarization or even in a multi-document summarization with a predefined domain. The only problem that has a great effect is the complexity of running the SVD. Therefore, the solution we propose is to reduce the dimensions of the original matrix before running the SVD, and this can be accomplished by integrating well-known and less time-consuming techniques (Jaccard Coefficient and VSM) with the LSA. The Jaccard similarity will process the sentences that share significant portions of the text, the VSM will process the sentences that

have statistically closest values (cosine similarity values), and the remaining sentences can then be processed by the LSA to solve the synonyms problem. Thus, we can see the MLS as a reduction of the original terms-sentences (or more general terms-documents) matrix that allows us to run the LSA for a larger number of sentences in a reasonable time interval.

Definition 8. Given $i\ x\ j$ original matrix X, let $Jacj_{red}$ be the number of sentences omitted by the Jaccard extraction, and let $VSMj_{red}$ be the number of sentences omitted by the VSM extraction, then the new j dimension j_{red} of matrix X will be

$$j_{red} = j - (Jacj_{red} + VSMj_{red})$$

Definition 9. Given $i\ x\ j$ original matrix X, let $Jaci_{red}$ be the number of terms omitted by the Jaccard extraction, and let $VSMi_{red}$ be the number of terms omitted by the VSM extraction, then the new i dimension i_{red} will be

$$i_{red} = i - (Jaci_{red} + VSMi_{red})$$

Example, for document 22, X contains 691 terms in 45 sentences, $Jaci_{red} = 182,\ VSMi_{red} = 276, then\ i_{red} = 691 - (182 + 276) = 233$ and, $Jacj_{red} = 11,\ VSMj_{red} = 17, then\ j_{red} = 45 - (11 + 17) = 17$, this means the input matrix to the SVD will be $X_{233\ x\ 17}$ $(MLS\ LSA)$ instead of $X_{691\ x\ 45}$ $(Classical\ LSA)$.

The SVD can be applied over the reduced MLS matrix $X_{i_{red}\ x\ j_{red}}$, because $0 < i_{red} \leq i$, and $0 < j_{red} \leq j$, (note that both of j_{red} and j_{red} are greater than zero because both the Jaccard similarity extraction and the VSM similarity extraction extracts return at least one sentence from the document). And, the reduced SVD produces $X_q\ where\ q \leq r$.

$$X_q = U_q . S_q . V_q^T \qquad (10)$$

The computation of the similarity between any two sentences $\boldsymbol{S_i, S_j}$ in the MLS extraction takes in consideration the similarities that are computed in the Jaccard extraction and the VSM extraction and can be viewed as follows:

$$sim(S_i, S_j) = \begin{cases} \dfrac{|S_i \cap S_j|}{\min\limits_{S_i, S_j \neq \emptyset} (|S_i|, |S_j|)}, & S_i, S_j \in X \\ \dfrac{\overline{S_i} . \overline{S_j}}{|\overline{S_i}| . |\overline{S_j}|}\ S_i, S_j \in X, if\ Jaccard\ Sim < 0.5 \\ \dfrac{\overline{S_i} . \overline{S_j}}{\|\overline{S_i}\| . \|\overline{S_j}\|}\ S_i, S_j \in S_q . V_q^T, if\ Jac, VSM\ Sim < 0.5 \end{cases} \qquad (11)$$

As mentioned in [32, 59], the complexity of the execution of the SVD in the classical LSA is $O(min(td^2, t^2 d))$, where t is the number of rows and d is the number of columns. In MLS extraction, the number of terms (rows) reduced from $i\ to\ i_{red}\ where\ i_{red} \ll i$ and the number of columns (sentence) reduced from $j\ to\ j_{red}\ where\ j_{red} \ll j$, this yields a complexity of $O(min(i_{red} {j_{red}}^2, {i_{red}}^2 j_{red}))$. The difference between t and i_{red} and d and j_{red} is significant, for example, for the document 22, the t value was 691 and i_{red} value was 233, the d value was 45 and j_{red} value was 17.

3.2 Algorithm Design and Description

This section describes in details the main steps and parameters used in our extraction system. The algorithm restructures each document as a set of documents. Each sentence represents a document. Unlike the algorithms that were implemented in [18, 24], our algorithm does not use the centrality as a feature that adds weight to the sentence score. We employ more sophisticated statistical and semantic techniques to compute the centrality. Also, the algorithm makes a recursive similarity calculation without the user or structure intervention. This deregulation makes the algorithm more flexible to generate unfocused generic extract.

To test our method of extraction, we build four entirely separated extraction systems, the first one is called the JacExtractor that is based on the Jaccard coefficient to measure the overlapped terms between two sentences. The second is called the VSMExtractor that is based on a tf.idf scheme to calculate the Cosine Similarity. The third system is called the LSAExtractor that investigates the semantic meaning behind the sentence and extracts the semantically related sentences. The LSAExtractor represents the employment of the classical LSA in Text Mining. The last extraction system is the MLSExtractor that is based on the MLS extraction model. After implementing these systems, we experimented them and collected the results for comparison.

Our design includes three stages: it starts by pre-processing the text. Next, the necessary parameters are computed and the similarity equations are applied. And, the final step involves the initiation of the deletion process that discards individual sentences based on the similarity calculations that are computed in the previous stage. In contrary to the bushy path and aggregate similarity, our algorithm considers the high similarities, discards low similarity values, and establishes one to one relationship between each pair of sentences. The following stages detail the main steps that are implemented in the design of the MLS method.

3.2.1 Stemming and Pre-processing

The pre-processing stage is essential to prepare the text for extraction. As presented in Fig. 1, the pre-processing includes

the representation of the document's sentences as sets of stems. It also includes filtering the text from Stopwords, punctuation marks, and strange words.

In our experiments, we used the Arabic Language as a domain language, and we employed the Khoja stemmer to find the stem for each term in the corpus. The Khoja stemmer is a well-known and significant stemmer for the Arabic language [62] and its popularity is equivalent to the popularity of the Porter stemmer for the English language [63]. The importance of the employment of the Khoja stemmer appears clearly in accelerating our process.

The preprocessing stage starts by dividing each document to a set of sentences and each sentence to a set of terms. The sentences are numbered sequentially according to their appearance in the document. The terms sets represent the input to the Khoja stemmer. Khoja stemmer produces stem for each term in the document and the output could be ROOT, NOT STEMMED, STOPWORD, NOT LETTER, PUNCTUATIONS, or STRANGEWORD. Therefore, the preprocessing stage is essential to produce the stems for the terms and to eliminate the STOPWORD, NOT LETTER, and PUNCTUATIONS. The following code summarizes the steps in the preprocessing stage:

Decompose d to a set d = {S1, S2, S3… Sn}
For each Si in d
Tokenize Si and extract its terms t1, t2… tk
For each tr in Si
Find the stem of tr using Khoja stemmer
If tr is Stopwords, Punctuation letter, and foreign terms then delete tr

3.2.2 Computing the Similarity

The similarity computations that are applied in the JacExtractor, the VSMExtractor, and the LSAExtractor are performed between each sentence in the document and all other sentences in the same document. We used the Jaccard coefficient, the cosine similarity based on tf-idf, and the cosine similarity based on LSA analysis in the similarity computation stage and we applied them in two situations, with and without MLS technique.

Part1: Similarity calculation procedure without employing the MLS: In this part, we applied the three similarity methods separately, and we collected the results in three different matrices. The JacSim matrix collects the results of applying the Jaccard similarity, the VSMSim matrix collects the results of applying the VSM similarity, and the LSASim matrix collects the results of applying the LSA similarity. The procedure takes the following detailed steps:

Assume that d is a set of sentences composing certain text document, t is the number of terms in d, n is the number of sentences, $X_{t\,x\,n}$ is our original matrix, and S_i and S_j are any two sentences in d.

Construct three (n x n) matrices: JacSim, VSMSim, and LSASim.
For each pair of sentences Si, Sj
Fill JacSim (i , j) using equation 2.
Fill VSMSim (i , j) using equation 4
Fill LSASim (i , j) using equation 9

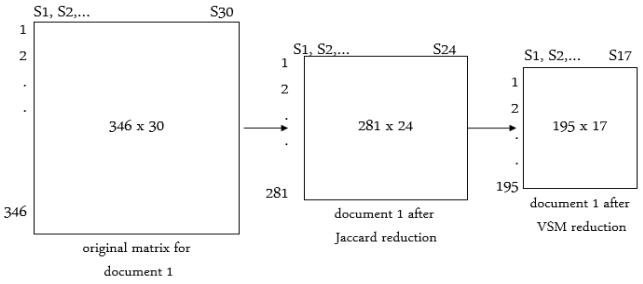
The output of this stage is three symmetric matrices with main diagonal values equal one. Each cell represents the similarity value between any two sentences. The procedure detailed above is greedy and the required complexity -in terms of space and time- is high because of the intensive runs of the LSAExtractor as described in section 3.1. Therefore, we suggested the MLS method in part2 of our experiment.

Part2: Similarity calculation procedure by employing the MLS: Fig. 1 shows the structure of the MLS extraction, after making the pre-processing stage, the system creates a terms-sentences matrix $X_{i\,x\,j}$, and this matrix will be the input for the JacExtractor, the VSMExtractor, and the SVD factorization subsystem. The SVD factorization subsystem reduces the matrix dimensions and produces X_r that will be the input of the LSAExtractor. At this moment, three extracts will be generated, one from each extractor (the dashed line in Fig. 1). Another two generated output from the JacExtractor and the VSMExtractor are: (1) the similarity values greater than 0.5 that go directly as input to the MLSExtractor, and (2) a list of sentences omitted by JacExtractor and VSMExtractor that helps to identify the new dimensions of the MLS reduced matrix by applying the equations that are mentioned in definition 8 and definition 9. Then, the reduced MLS matrix X_q is processed by the SVD factorization system and equation 11 is applied to generate the similarity matrix for the MLSExtractor system. The MLSExtractor receives the similarity values coming from three sources, the JacExtractor, the VSMExtractor, and the SVD factorization subsystem (this represents the application of equation 11) and merges them to produce a single similarity matrix.

In abstract words, the method starts by computing the Jaccard coefficient and decides whether more similarity computation is necessary or not. If the Jaccard similarity between two sentences is high (greater than 50%), this implies that the two sentences shared sufficient portion of text and no further calculations are needed. For example, the Jaccard between the following pairs of sentences from document1 ((S1, S4), (S1, S9), (S1, S10), (S5, S6), (S5, S8), (S8, S11), (S9, S16), (S23, S27))

exceeded 50%. Thus, no further similarity computation is required for these pairs of sentences, and all the LSA similarity computation of the sentences S4, S6, S8, S9, S10, and S27 will be omitted.

From our example on document 1, the dimensions of the original matrix X that the classical LSA will start from are 30x346 (30 sentences, and 346 terms), and after applying the MLS method, these dimensions become 17x195. The number of runs of the Si, Sj similarity computations reduced from 435 in classical LSA extraction to 241 in the MLS extraction (reduced the number of runs of the LSA similarity by 45%). (For complete results see Fig. 11 and 12 in section 6.4). The form of the MLS matrix reduction for document 1 looks like the following:



As we discussed previously, we want to reduce the number of runs of the LSA similarity, and also to reduce the storage requirement. Thus, we amend the previous procedure appeared in part 1 to be as follows:

Assume that d is a set of sentences composing certain text document, t is the number of terms in d, n is the number of sentences, $X_{t\,x\,n}$ is our original matrix, and S_i and S_j are any two sentences in d.

Construct two (n x n) matrices: JacSim, MLSim
for each pair of sentences S_i and S_j in $X_{t\,x\,n}$
fill JacSim(i,j) using equation 2.' The Jaccard will be the first step with or without MLS
$i_{red} = t\,,\ j_{red} = n$
for each entry in JacSim matrix
if JacSim(i,j)>50%
MLSim(i,j)=JacSim(i,j) and Delete S_j column and rows(for each t in S_j) from $X_{t\,x\,n}$
$i_{red} = i_{red} - number\ of terms\ in\ S_j\ , j_{red} = j_{red} - 1$
run the VSMExtractor over the $X_{t\,x\,n}$,
for each S_i and S_j not found in MLsim(i,j)
If the VSM Similarity (i, j)>50%
MLsim(i,j)=VSM(i.j) and Delete S_j column and rows(for each t in S_j) from $X_{t\,x\,n}$
$i_{red} = i_{red} - number\ of terms\ in\ S_j\ , j_{red} = j_{red} - 1$
Construct new terms-documents matrix $X_{i_{red}\,x\,j_{red}}$
run the LSAExtractor over $X_{i_{red}\,x\,j_{red}}$
MLsim(i,j) = LSASim(i,j)

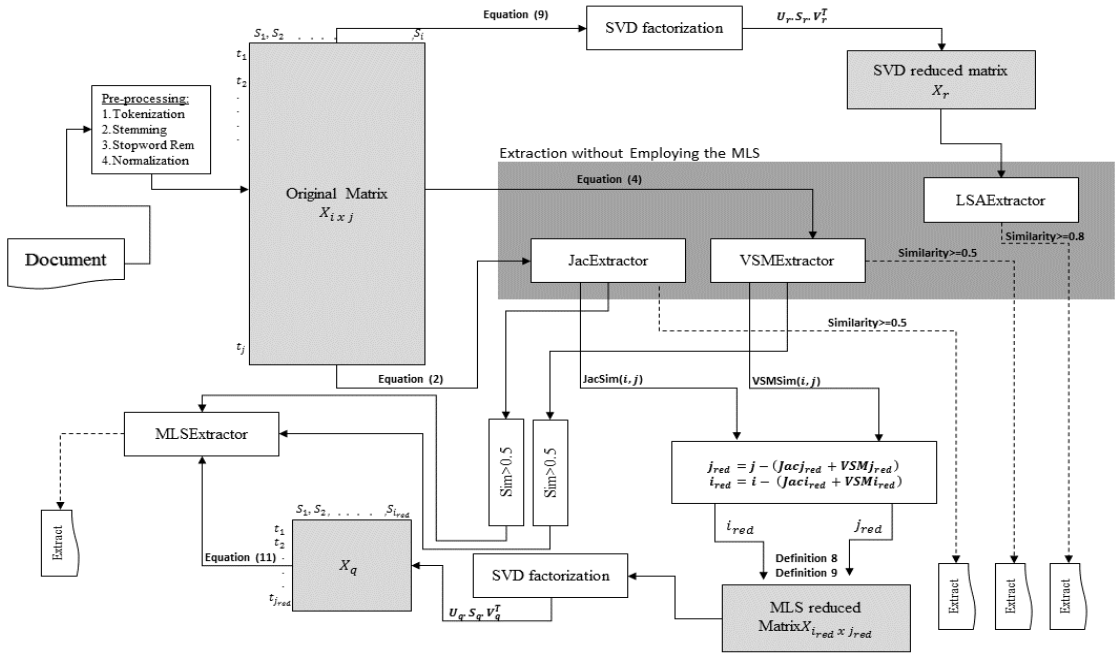


Fig. 1 the MLSExtractor Architecture

This pseudo-code implements definition 8 and 9 to obtain the reduced matrix dimensions, and equation 11 to construct the similarity matrix. Note that each time the algorithm finds a large value of the Jaccard or the VSM cosine similarity the corresponding column and rows of the similar sentence are removed from the original matrix $X_{t\,x\,n}$. Accordingly, when applying the LSAExtractor at the end of this algorithm, the original matrix is reduced to $X_{i_{red}\,x\,j_{red}}$. It appears clearly that the part of the algorithm that consumes a lot of time is confined and used with a small number of input data (terms and sentences).

3.2.3 The Deletion Process

The deletion algorithm is the core of this research; it is a recursive procedure that investigates the diversity among the sentences. The algorithm gives all the sentences found in the text being summarized the same value of importance without bias to structural, linguistic, or domain features. Therefore, our algorithm can be applied to one document, multi-document, or any piece of text that contains two sentences or more.

The proposed deletion process in this research is the process of discarding certain sentence(s) based on two parameters:

Parameters Identification: The first parameter is *the generated similarity value* in the second stage. This value decides whether the corresponding sentences will be deleted or not if they exceeded the threshold value. The second parameter that the deletion process considered is *the existence of a base sentence* S_i. The algorithm deletes S_j if the similarity between S_i and S_j exceeded the threshold value and S_i was not deleted before otherwise, S_j remains. We cannot remove S_j if the similar sentence Si was already removed. Thus, the condition that decides whether the sentence will be deleted or not looks like the following:

$$if\ the\ Similarity\ between\ (S_i, S_j) \geq threshold\ value \\ and\ S_i \notin Deleted\ list, then\ delete\ S_j$$

Setting the similarity Threshold Value: Before discussing the threshold values, we should define the RSI (Ratio of Sentences Intersection) and the CR (Compression rate) because both of them are used to determine the threshold value and to evaluate the accuracy of the generated extracts.

Definition 10: Let A = {S_1, S_2, …, S_n} be a set of sentences in the automatic extract that is generated by one of our automatic extraction systems for document d_i. And, let M = {S_1, S_2, …, S_m} be a set of sentences in the reference extract for document d_i, then

$$RSI(A, M) = \left(\frac{|A \cap M|}{\min(n, m)}\right)(100\%) \ldots . (12)$$

Example: for document 1 the sentences found in the manual extract 4 (M4) was 2, 5, 7, 8, 13, 15, 17, 29, and the sentences extracted using the MLSExtractor was 1, 2, 5, 7, 14, 15, 17, 18, 20, 22, 26, and 29.

$\text{RSI (M4 extract, MLS extract)} = \left(\frac{6}{8}\right)(100\%) = 75\%$ (For document1, 75% of the sentences generated in M4 extract appeared in the MLS extract). The RSI value is range between 0 and 100%. The value 100% for the RSI means that all the reference summary sentences were found in the automatic extract.

Definition 11: Let t be the number of terms in document d, and t1 is the number of terms in the extract e:

$$CR(e) = \left(\frac{t1}{t}\right)(100\%) \ldots (13)$$

Example: for document 1: t = 414 and the number of terms found in the extract that is generated by the MLSExtractor was 150 , the $\text{CR} = \left(\frac{150}{414}\right)(100\%) = 36\%$. The CR value is range between 0 and 100%. The value 100% for the CR means no compression happened.

To specify the threshold values, we generated the similarity matrices for a sample of 13 documents. The selected documents have a variance number of sentences, for example, document 1 contains 30 sentences, document 2 contains 15 sentences, and document 3 contains 4. The first look on the similarity matrices of the documents in our sample gave us an indication of the possible threshold values. For the VSMExtractor and the JacExtractor, we tested three possible ranges: greater than 25%, greater than 50%, and greater than 75%. The test included the computation of the RSI and the CR for the documents at the three threshold values (Fig. 2.a and 2.b presented the results). The threshold value greater than 25% produced low values of RSI 40% and 36%. The threshold value greater than 75% yielded high values for the RSI, but no compression was done (CR values were above 81%). The threshold value greater than 50% produced reasonable RSI results (67%, 84%) and the CR value is moderate in size, but still, it is better than 25% and 75%.

For the LSAExtractor, the LSA similarity calculations investigate the semantic similarity between the document sentences, and the obtained similarity values from the second stage were high because all the sentences belonging to one document are talking about one topic. Therefore, we raised the threshold value, and we tested three threshold values: greater than 70%, greater than 80%, and greater than 90%. The RSI and CR values were computed and presented in Fig. 2.c, the selected threshold value was “greater than 80%”.

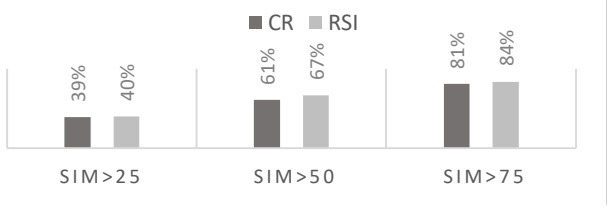


Fig. 2.a the Average CR and the Average RSI at Three Threshold Values of the VSM Similarity Values.

Fig. 2.b the Average CR and the Average RSI at Three Threshold Values of the Jaccard Similarity Values.

Fig. 2.c the Average CR and the Average RSI at Three Threshold Values of the LSA Similarity Values.

Deletion Process Implementation and Complexity: The implementation of the deletion process in the JacExtractor, the VSMExtractor, and the LSAExtractor is very simple. The three extractors test the parameters that are described above against the specified threshold values. But, the implementation of the deletion process in the MLSExtractor is a bit different because the generated similarity matrix MLSim in section 3.2.2 part 2 is a merged matrix contains similarity values coming from three sources as specified in Fig. 1. Thus, some of these values should be tested against 0.5 threshold value (the ones come from the Jaccard and the VSM similarity calculations), and the others should be tested against 0.8 threshold value (the ones come from the LSA similarity calculation). For this reason, we structured the sentence number in the similarity matrix as a pair of number and identifier, the number represents the sentence number, and the identifier takes a binary value, 0 indicates the VSM or the Jaccard similarity and 1 indicates the LSA similarity.

Assume that Del contains the deleted sentences and recursively more sentences will be placed in Del as the deletion process precedes to the end. The deletion of the sentence in the JacExtractor and the VSMExtractor constraints by the condition: $SM(S_i, S_j) \geq 0.5 \; and \; S_i \notin Del$, and the deletion of the sentence in the LSAExtractor constraints by the condition: $SM(S_i, S_j) \geq 0.8 \; and \; S_i \notin Del$, whereas the deletion of the sentence in the MLSExtractor constraints by the condition: $SM(S_i, S_j) \geq 0.5 \; and \; S_i \notin Del$ in the first and second layers and the condition: $SM(S_i, S_j) \geq 0.8 \; and \; S_i \notin Del$ in the third layer. The pseudo-code of the deletion algorithm in the MLS extraction is as follow:

Input: MLSim (j x j); where j is the number of sentences
Output: list of extract sentences id's E, a list of deleted sentences id's Del
for each pair of sentences S_i *and* S_j *in*
if $i \neq j$ *then*
if j.identifier = 0 {
if MLSim (i x j)>0.5 and $i \notin Del$
add i to E and add j to }
else
if MLSim (i x j)>0.8 and $i \notin Del$
add i to E and add j to Del

The input of the deletion algorithm will be the $X_{n\,x\,n}$ matrix where n is the number of sentences. $X_{n\,x\,n}$ has $\left(\frac{n x\, n}{2}\right)$ elements (and $\left(\frac{n\,x\,n}{2}\right)$ are empty) and the diagonal values are ones representing the $sim(S_i, S_i)$ values. Thus, in this process, we need n-1 comparisons to process the first sentence. The process of the second sentence requires n-2 comparisons because we already accomplished the similarity computation between the first and second sentences, and the third sentence needs n-3 and so on. This means that the overall complexity can be shown as:

$$Number\ of\ comparisons = (n-1) + (n-2) + \cdots + 1$$

$$= \frac{n\,(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} = \frac{1}{2}(n^2 - n)$$

This means that the algorithm yielded $O(n^2)$ to process n sentence. However, the number of sentences is practically not very large and this makes the application of the deletion process is acceptable.

4 Experiment

This section describes the conducted experiment and procedure to measure the effectiveness of our extraction technique. It was designed to collect the results that are generated by applying the proposed deletion process in the extraction systems. The data preparation, the environment setup, and the experimental procedure are explained in details in the next subsections.

4.1 Data Preparation

At first, our experiment to test the effectiveness of our technique was applied to three well-known datasets for the Arabic Language.

Essex Arabic Summaries Corpus: This corpus is published on the Lancaster University website[5]. The corpus contains 153 Arabic articles and 765 human-generated extractive summaries. For each document, there are five manual extracts. The corpus contains documents with different subject areas including art, music, science, technology, education, finance, health, politics, and religion. The corpus used recently by Al-Radaideh and Bataineh in [37].

Kalimat[6] Corpus, the corpus contains a huge number of documents and their automatic and manual summaries. It contains 20,291 Arabic documents, with 20,291 automatically generated summaries.

Another corpus used in our experiment was 242 documents used in [60,67], ten chapters were taken from the corpus, and the purpose was to test our technique for large documents.

4.2 Environment Set up

The experiment was performed on Intel® Core™ i5-7200U CPU @ 2.5GHz processor with 8 GB RAM and Windows 10 OS. We used VB 2013 programming language to implement our Extractors, with Excel sheets as an interface. To make our experiment more accurate, we linked our software with the Latent Semantic Analysis Software developed by the University of Colorado Boulder. Also, we used Khoja stemmer to produce the Arabic stem for each term in the documents.

4.3 Experiment Procedure

After implementing the extraction systems, we conducted our experiment procedure and we followed the following steps:

Step1: Data pre-processing that took the following detailed steps:

(1) Load each document to Khoja stemmer software and generate a list of stems.
(2) From the generated list remove Stopwords.
(3) From the generated list remove punctuation symbols.
(4) From the generated list remove non-letter symbols.
(5) From the generated list remove foreign words.
(6) Replace the Arabic letters { ء , إ , أ , آ} by ا for normalization.
(7) Replace the Arabic letter ة by ه for normalization.
(8) Replace the Arabic letter ئ, ى by ي for normalization.
(9) Remove diacritics.

After applying the pre-processing stage, the generated sentences and terms are organized in excel sheets. These sheets contain the required parameters with their values that are necessary to initiate the second stage described in the previous section.

Step2: Processing the sheets using JacExtractor, VSMExtractor, LSAExtractor, and MLSExtractor

The data in the sheets were processed by our extractors, and the generated similarity values were organized in two-dimensional matrices. Table 2 shows the matrix template that was used to collect the similarities among the document sentences. The x's characters represent the similarities values, where $0 <= x <= 1$. Four different matrices from the four extraction systems were generated for each document.

Table 2 the Template of the Sentences Similarity Matrix

S1	**X**	**x**	**x**	**x**	**x**	**x**
S2		**x**	**x**	**x**	**x**	**x**
S3			**x**	**x**	**x**	**x**
.				**x**	**x**	**x**
.					**x**	**x**
.						**x**
Sn	S2	S3	.	.	.	Sn

Step 3: Selecting certain sentences to be deleted (the application of the deletion process).

In this step, we initiated the deletion process. We investigated each value of x appeared in Table 2 and decided to delete or not the corresponding sentence depending on the conditions that are explained in Section 3.

Before going to the last step, we present a practical example. The example practices the deletion process used in the all automatic extractors designed in this paper. The similarity calculations appear in this example is computed using the VSMExtractor for document 17. Table 3 details the attributes of document 17.

In this example, we present in details the deletion steps of the redundant sentences of document 17, this document contains a moderate number of sentences (14) and words (410), and the document subject (Environment) is general subject. However, we can replace document 17 with any document.

Table 3 Attributes of Document 17

Attribute	Value
Document Subject	Environment
Source	Essex Corpus
Title	Chemistry (الكيمياء)
Number of sentences	14
Number of words	406

[5] http://www.lancaster.ac.uk/staff/elhaj/corpora.htm

[6] Can be downloaded from: https://sourceforge.net/projects/kalimat/

After computing the necessary parameters, we can apply equation 4 through the first and second stage in our algorithm and fill up the similarity matrix as shown in Table 4. The intersection cells values represent the cosine similarity values. For example, the similarity between sentence 1 and sentence 3 is 0.88, and between sentence 7 and sentence 13 is 0.20 and so on.

From Table 4, we can delete certain sentences depending on the similarity values. Table 4 shows that the sentences 1, 3, 5, 9, and 13 have high similarity values, which means that we can delete the sentences 3, 5, 9, and 13. The sentences 2, 3, and 4 have high similarity, but 3 is deleted in the previous step so we can delete only sentence number 4. Similarly, the sentences 6 and 9 have high similarity (0.77), but 9 was deleted before. The similarity value between 7 and 8 is greater than 0.5, and sentence 8 will be deleted. The similarity between 13 and 14 is greater than the threshold value, but we cannot delete sentence 14 because we already deleted sentence number 13 (the second condition with sentence number 14 in our algorithm will be false). The remaining sentences that will appear in the automatic extract are st1, st2, st6, st7, st10, st11, st12, and st14. See Table 5.

Table 4 VSM Similarity Matrix for Document 17

	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
S1	0.43	0.88	0.43	0.80	0.26	0.29	0.06	0.98	0.44	0.07	0.26	0.54	0.26
S2		0.76	0.75	0.00	0.13	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.45
S3			0.77	1.00	0.33	0.05	0.42	0.31	0.00	0.01	0.32	0.07	1.00
S4				0.70	0.26	0.20	0.19	0.15	0.06	0.09	0.16	0.03	0.62
S5					0.78	0.20	0.04	1.00	0.00	0.05	1.00	0.26	0.46
S6						0.01	0.27	0.77	0.07	0.22	0.08	0.02	0.32
S7							0.83	0.43	0.44	0.37	0.44	0.20	0.02
S8								0.27	0.18	0.40	0.10	0.05	0.25
S9									0.12	0.10	0.38	0.06	0.10
S10										0.31	0.13	0.00	0.00
S11											0.18	0.05	0.29
S12												0.06	0.10
S13													1.00

Table 5 Document 17 Similarity Matrix after Deleting Sentences 3, 4, 5, 8, 9, 13.

	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
S1	0.43	0.88	0.43	0.80	0.26	0.29	0.06	0.98	0.44	0.07	0.26	0.54	0.26
S2		0.76	0.75	0.00	0.13	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.45
S6						0.01	0.27	0.77	0.07	0.22	0.08	0.02	0.32
S7							0.83	0.43	0.44	0.37	0.44	0.20	0.02
S10										0.31	0.13	0.00	0.00
S11											0.18	0.05	0.29
S12												0.06	0.10

Step 4: Extract Generation.

The sentences remaining after applying the deletion process are concatenated sequentially according to their order in the original document.

5 Extraction Results

After implementing the deletion process in the automatic extraction systems and initiating the experiment, we collected our results to make further analysis. For each document, we collected the following:

The sentences composing the automatic and the manual extracts: For each document, the sentences that were generated from the VSMExtractor, the JacExtractor, the LSAExtractor, the MLSExtractor, and the sentences that forming the manual extracts are collected in a table as in Table 6. The document sentences were numbered sequentially during the pre-processing stage.

Table 6 the Sentences Composing the Automatic and the Manual Extracts for Document 1

Extract	List of Sentences													
JAC	1	2	3	5	7	11	12	13	14	15	16	17	18	19
VSA	1	2	3	5	7	9	14	15	17	18	19	20	21	22
LSA	1	2	4	5	6	9	10	12	13	15	16	18	22	24
MLS	1	2	5	7	14	15	17	18	20	22	26	29		
M1	3	7	12	30										
M2	2	5	7	8	13	15	17	29						
M3	2	3	5	21										
M4	2	5	7	8	13	15	17	29						
M5	2	3	5	21										

The Compression Rate: For each automatic extract, we computed the length of the extract relative to the length of the document (CR computation using Definition 11). See Table 7 as an example.

Table 7 Compression Rates Samples

doc #	JacExtractor	VSMExtractor	LSAExtractor	MLSExtractor
1	81%	56%	55%	55%
2	100%	45%	40%	36%
3	86%	92%	42%	35%

We used the CR to estimate the size of the automatic and the manual extract. This estimation together with the RSI measure will give us a clear indication of the effectiveness of our similarity calculations and the deletion process used in our extraction systems.

The values of RSI: For each automatic extract we computed the RSI value from equation 12 in definition 10: Table 8 gives an example of the RSI values for document 57, 135, 136. RSI will be used later in our evaluation to estimate the ratio of containment (see definition 12) of the manual extracts in our automatic extracts.

Table 8 Samples of RSI Values between the JacExtractor Extracts and the Manual Extracts (M1-M5) Taken from Essex Corpus

doc#	(M1,Jac)	(M2,Jac)	(M3,Jac)	(M4,Jac)	(M5,Jac)	AVG
135	0%	31%	31%	40%	71%	35%
57	54%	67%	40%	0%	67%	45%
136	54%	33%	67%	0%	75%	45%

The definition of RSI comes in section 3.2.3, and as we said before, the RSI measures the percent of complete sentences shared between two extracts, but to make the RSI more significant, we categorized the RSI values in ranges as shown in definition 12.

Definition 12: Let Sim_x be the value of RSI between the automatic and the manual extract:

LOW Containment occurs if $Sim_x < 50\%$.
MODERATE Containment occurs if $50\% <= Sim_x < 75\%$.
HIGH Containment occurs if $75\% <= Sim_x < 100\%$
FULL Containment occurs if $Sim_x = 100\%$.

Example: For document 6, the RSI (M1 extract, MLS extract) was 100%, this means that all the sentences found in the manual extract M1 appeared in the MLS extract and this yields FULL Containment.

We found the ratio of the automatic extracts that obtained LOW, MODERATE, HIGH, and FULL Containments with the manual extracts. For example, in Table 9, the FULL-Containment value between the automatic extracts that were generated using the VSMExtractor and the manual extracts was 16%. This means that 16% of these automatic extracts contained all the sentences of the manual extracts.

Table 9 the Containment of Manual Extracts in VSMExtractor Extracts

Containment (VSMExtractor Extract, manual extracts)	
LOW	19%
MODERATE	37%
HIGH	29%
FULL	16%

The values of Avg_Recall, Avg_Precision, and Avg_F-score using Rouge 2.0: For each automatic extract that was generated by our automatic extractors and by two existing automatic extractors (UTF-8 support tool, API summarizer), we used Rouge 2.0 tool to find the *Avg_Recall, Avg_Precision, and Avg_F-score* between the automatic extracts and the manual extracts that are taken from Essex and Kalimat datasets; Table 10 shows an example of Rouge results for Document 29.

Table 10 Rouge Results for Document 29 from Essex Corpus

Extraction System	Avg_Recall	Avg_Precision	Avg_F-Score
API summarizer	0.38	0.82	0.51
LSA Extractor	0.68	0.61	0.64
JacExtractor	0.48	0.43	0.45
UTF-8 SUPPORT TOOL	0.31	0.55	0.40
MLS Extractor	0.87	0.62	0.72
VSM Extractor	0.76	0.55	0.64

6 Evaluation and Analysis

The evaluation of the quality of the automatically generated extracts was performed using the values of the Avg_Recall and Avg_Precision (generated from Rouge 2.0 evaluation tool) and the values of RSI that were integrated with the values of CR. After collecting the automatic extracts from the developed extractors, we compared the results of our automatic extractors with the results that were generated from two existing multilingual automatic extraction systems, UTF-8 SUPPORT tool, and Text Summarization API tool. The same documents that had been processed by our extractors were processed using the two existing extractors, and the recall, precision, and f-score values were collected. The final step of our evaluation was to analyze the time consumed by each extraction systems and to measure the achieved enhancement by the MLS method on both the matrix reduction and the number of runs of the LSA procedure.

6.1 The Containment Evaluation

6.1.1 RSI Findings

We used the RSI to measure the ratio of containment of manual extracts in the automatic extracts. The Figures from Fig.3 to Fig.6 show the ratio of containment of the Essex manual extracts (five manual extracts for each document) in the automatic extracts that were generated from the JacExtractor (Fig.3), the VSMExtractor (Fig.4), the LSAExtractor (Fig.5), and the MLSExtractor (Fig.6).

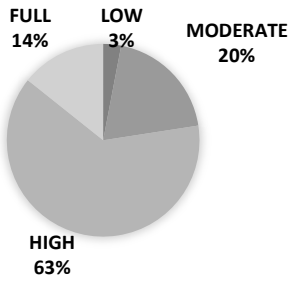


Fig. 3 the Containment of the Manual Extracts in the JacExtractor Extracts

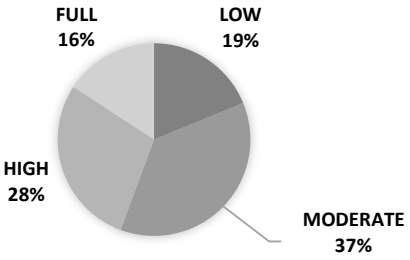


Fig. 4 the Containment of the Manual Extracts in the VSMExtractor Extracts.

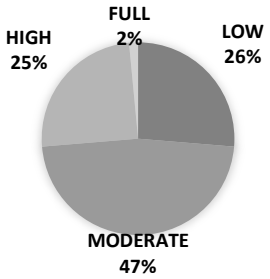


Fig. 5 the Containment of the Manual Extracts in the LSAExtractor Extracts.

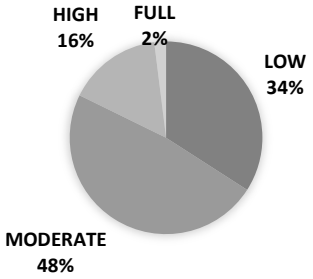


Fig. 6 the Containment of the Manual Extracts in the MLSExtractor Extracts.

From Fig. 3- 6 we note the following:

(1) The LOW Containment means that less than half of the sentences found manually appeared in the automatic extract. The obtained LOW Containment values by our extraction systems were low ranging from 3% in the JacExtractor and 34% in the MLSExtractor. The LOW Containment value gives a good indication of the extracting accuracy because their values strongly determine the contents of the automatic extract. In fact, when the value of the LOW Containment is 34% in the MLSExtractor extracts, this means that 66% of the manual extract sentences appeared in the automatic extracts. In other words, the vast majority of the MLSExtractor extracts contain more than 50% of the sentences of the manual extracts.

(2) The FULL and HIGH Containment values for both the JacExtractor and the VSMExtractor were high and greater than their corresponding values in the LSAExtractor and the MLSExtractor.

(3) The MODERATE plus HIGH Containment (the containment of greater than 50% and less than 100% of the manual extracts in automatic extracts) dominated the largest sector for the four extraction systems. The value of the MODERATE-plus HIGH Containment was 85% for the VSMExtractor, 83% for the JacExtractor, 72% for the LSAExtractor, and 64% for the MLSExtractor.

(4) The value of the FULL plus HIGH Containments (the containment of greater than 75%) for the JacExtractor and VSMExtractor were somewhat high 66% and 44%. Whereas, the FULL and HIGH Containment value was acceptable (27%) for the LSAExtractor and low for the MLSExtractor (18%). The percent of 100% Containment (FULL-Containment) was noticeable for the JacExtractor and VSMExtractor, 14% and 16% respectively, but for the MLSExtractor and the LSAExtractor, the FULL Containment appeared in two cases only and showed 2% FULL Containment.

6.1.2 Integrating RSI and CR

If we consider the FULL and HIGH Containment as the optimal results, we can order the accuracy of our automatic extractors as follows: The VSMExtractor comes at the top followed by the JacExtractor followed by the LSAExtractor, and the MLSExtractor comes at the end. But, the RSI Containment as a measure of evaluation is helpful if we combine it with the Condensation Rate, which is an important measure used to test the quality of the machine extracts.

If we integrated the results showed in Fig. 3-6 with Fig. 7, we find that the RSI Containment values for both the JacExtractor and the VSMExtractor extracts were obtained at a high value of CR (79% and 68% respectively), and this explains why the JacExtractor and the VSMExtractor extracts contained more sentences of the manual extracts than the LSAExtractor and MLSExtractor extracts. From this point of view, we can find that the LSAExtractor system surpassed the JacExtractor and the VSMExtractor because it obtained high Containment value in a reasonable CR value (54%). Regarding the MLSExtractor, it achieved reasonable Containment assessment (66%), and it succeeded to remove 58% of the original text. Fig. 7, reveals the major drawback of the VSMExtractor and the JacExtractor systems. Their CR values are impractical, which means that these systems did not cancel a generous portion of the original text. On the other hand, the LSAExtractor and the MLSExtractor systems roughly decreased the text size to the half and obtained significant RSI values.

Table 11 Rouge Results for Six Datasets (Essex Corpus)

	Average Recall	Average Precision	Average F-score
Education (5 docs)			
JacExtractor	0.63	0.25	0.36
VSMExtractor	0.61	0.26	0.36
LSAExtractor	0.50	0.21	0.29
MLSExtractor	0.54	0.29	0.37
Art – Music (10 docs)			
JacExtractor	0.59	0.27	0.37
VSMExtractor	0.50	0.29	0.37
LSAExtractor	0.69	0.38	0.45
MLSExtractor	0.38	0.31	0.33
Environment (30 docs)			
JacExtractor	0.70	0.36	0.46
VSMExtractor	0.68	0.37	0.47
LSAExtractor	0.68	0.42	0.51
MLSExtractor	0.55	0.41	0.46
Finance (14 docs)			
JacExtractor	0.59	0.33	0.43
VSMExtractor	0.72	0.32	0.44
LSAExtractor	0.63	0.41	0.48
MLSExtractor	0.57	0.43	0.44
Health (12 docs)			
JacExtractor	0.78	0.42	0.54
VSMExtractor	0.68	0.39	0.49
LSAExtractor	0.58	0.53	0.54
MLSExtractor	0.45	0.50	0.46
Politics (14 docs)			
JacExtractor	0.75	0.29	0.42
VSMExtractor	0.65	0.29	0.39
LSAExtractor	0.51	0.46	0.46
MLSExtractor	0.46	0.41	0.40

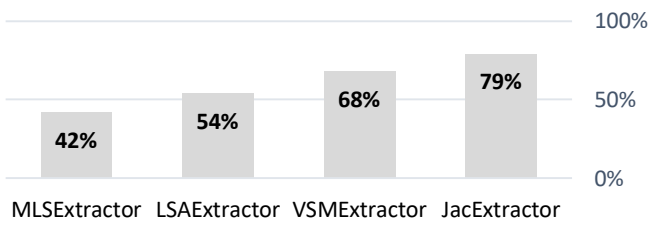


Fig. 7 the Average Compression Rates for the Extracts Generated by the Four Automatic Extractors.

6.2 Evaluation Using Rouge Evaluation Tool

We used Rouge 2.0 evaluation tool to measure the similarity between the automatically generated extracts and the manual reference extracts found in Essex and Kalimat datasets.

6.2.1 Rouge Evaluation over Essex Data Corpus

We used the same documents that are tested in the Containment evaluation, and we separated the documents to six datasets depending on their subject, and we collected the recall, precision, and F-score for each dataset and for all the datasets together. The datasets have a different number of documents, the smallest one contains five documents, and the largest dataset contains 30 documents. The purpose of the separation of the documents to smaller data sets is to evaluate the automatic extracts with a variance number of documents.

Fig. 8 and Table 11 show the final Rouge results for the documents taken from Essex corpus. Table 11 shows that for the six datasets, the MLSExtractor extracts obtained the lowest recall value. The average recall value of the MLSExtractor ranged from 38% in dataset2 with 10 documents to 55% in dataset3 with 30 documents, and for the whole corpus, the average recall for the MLSExtractor was 48% (as showed in Fig. 8).

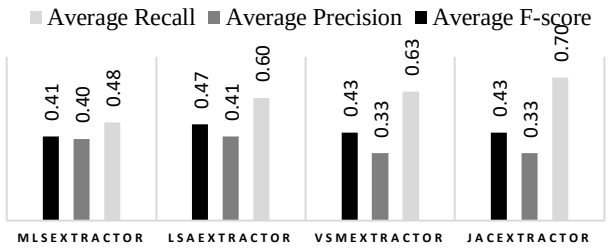


Fig. 8 Rouge Results for the Four Automatic Extraction Systems (Essex and 242-Document corpus)

From Fig. 7 and 8, the MLSExtractor extracts shared 48% with the manual extracts at 42% CR value, whereas the JacExtractor achieved 70% recall but at 79% compression rate. Note that the higher compression rate means the system failed to omit a large portion of the text. Regarding the precision values, the MLSExtractor obtained an average precision that was higher than the average precision of the VSMExtractor extracts and the JacExtractor extracts, and the average precision of the MLSExtractor was very close to the average precision of the LSAExtractor extracts.

Also, Fig. 8 shows that the LSAExtractor gave 41% average precision, which was greater than the average precision of the other automatic extraction systems. Note that the average precision value of the VSMExtractor and the JacExtractor are 33%, 33% with average CR values equal 68% and 79% respectively. The average recall of the LSAExtractor was less than the average recall of the VSMExtractor and the JacExtractor because both of the JacExtractor and VSMExtractor failed to remove a reasonable part of the text and this appeared clearly from their CR values. The CR value of the LSAExtractor is less than the CR value of the VSMExtractor by 14% and less than the CR value of the JacExtractor by 25% (see Fig. 7).

The obtained Rouge results from the MLSExtractor were very close to the LSA results, the average precision for the MLSExtractor is less than the average precision for the LSAExtractor by 1%, and the average F-score for the MLSExtractor are less than average F-score for the

LSAExtractor by 6%. However, the MLS extraction achieved those Rouge results at 42% CR rate, which gives the MLS extraction an advantage over the other automatic extraction.

6.2.2 Rouge Evaluation over Kalimat Corpus.

In this subsection, we established the same experiment over Kalimat data corpus. Fig. 9 shows the results, the achieved average recall, precision, and F-score values were higher than the ones that appear in Fig. 8 because in Kalimat corpus we have only one manual reference summary while in Essex we have five manual reference summaries and this reduces the number of comparisons that are performed by Rouge 2.0. Fig. 8 and 9 present convergent trends. Indeed, we neglected the precision and recall values related to the JacExtractor and the VSMExtractor because the corresponding CR was insignificant. In Fig. 9, the precision values were convergent, and the MLSExtractor obtained the highest (0.7) value, the recall of the LSAExtractor was higher than the recall of the MLSExtractor by 15% but with 12% difference in the CR (see Fig. 7). However, 57% of recall value is considered significant (see Fig. 14 for a comparison with the obtained recall values in other research papers) if we connect that with the CR and with the amount of reduction obtained in both, the original matrix and the number of runs of the LSA similarity function (see Fig. 12).

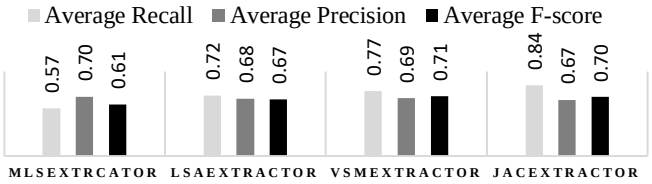


Fig. 9 Rouge Results for the Four Automatic Extraction Systems (Kalimat corpus).

6.3 Comparing MLS Extract with Existing Automatic Extraction Systems

The previous assessment of our extraction systems compares the automatically generated extracts with the manual extracts found in Essex and Kalimat datasets. In this subsection, we examined the results of our MLSExtractor with the results that were generated from two multilingual automatic extraction systems, the UTF-8 SUPPORT TOOL[7], and the Text Summarization API[8].

Text Summarization API is free software to extract the salient sentences from the original text by specifying the number of sentences needed. It is a Machine learning approach for extraction that can be used on different platforms.

UTF-8 SUPPORT TOOL is a feature based summarization system that examines certain features during the summarization process such as the sentence position, the centroid, keywords, and common subsequences. It is a single and multi-document summarization tool that supports multilingual summarization.

We used the UTF-8 SUPPORT TOOL and the API summarizers because they are multilingual summarizers that can be applied to our Arabic datasets. The same documents that were used during the evaluation of our automatic extraction systems were summarized using the API and the UTF-8 SUPPORT TOOL summarizers. The Rouge evolution tool is used to evaluate the generated extracts against the manual extracts, and the recall and precision values were collected. We used 40% CR for both UTF-8 SUPPORT TOOL and API Summarizers because we obtained this value of CR for the MLS extraction and this experiment compares the generated extracts from these two automatic systems with our MLS extracts.

The obtained average recall and precision for the three automatic extractors experimented in this section is shown in Fig. 10. From Fig. 10, we see that the recall for our automatic extractor was the highest (48%) followed by the UTF-8 SUPPORT TOOL followed by the API. The precision results were convergent, the precision of the MLSExtractor was 40%, which is 1% greater than the precision of the UTF-8 SUPPORT TOOL and 4% greater than the precision of API.

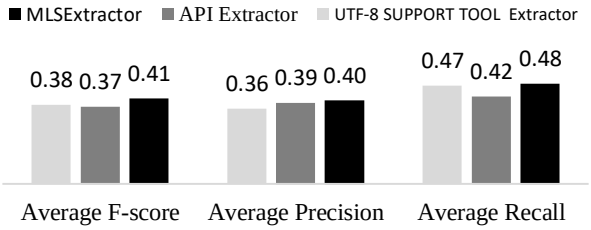


Fig. 10 Average Recall and Precision Values for MLS, API, and UTF-8 SUPPORT TOOL Extractors

The noted behavior from the collected recall and precision values for the UTF-8 SUPPORT TOOL and the API systems was the variances in the recall and precision values from document to document. Fig. 11.a and 11.b show the fluctuation of the recall and precision values over the first 86 documents. For the UTF-8 SUPPORT TOOL and API extractors, the lines that represent the recall and precision values swing far from

[7] https://www.tools4noobs.com/summarize/

[8] http://textsummarization.net/

their arithmetic mean. To accurately estimate the fluctuation, we computed the standard deviation of the resulted recall and precision values for the three systems. We employed the standard deviation to measure how close the individual values of recall and precision to their mean. If the value of the standard deviation is high, this means that the values are disparate. The standard deviation values came as followed:

Precision _STDEV (MLS EXTRACT) = 0.09
Precision _STDEV (API EXTRACT) = 0.16
Precision _STDEV(UTF-8 TOOL EXTRACT) = 0.1
Recall _STDEV (MLS EXTRACT) = 0.1428
Recall _STDEV (API EXTRACT) = 0.18210
Recall _STDEV (UTF-8 TOOL EXTRACT) = 0.2

Combining the average precision and recall with the standard deviation we can argue that the MLSExtractor surpassed the other two automatic extractors because it obtained the highest recall with convergent precision with the lowest standard deviation for both the recall and the precision values. The value of the standard deviation showed that in most cases the MLSExtractor worked in a stable way and gave recall and precision values that were very close to their mean.

6.4 Time Complexity Analysis

The Jaccard Coefficient matches the terms of two sets; each set represents one sentence terms, if we have *n* sentences in a given document and *m* terms in each sentence, the overall time complexity will be $O(mn)$. The VSM computes the term weight over the whole corpus, if we have *t* terms in the whole corpus and d document in the corpus, this means that we need $O(td)$ time complexity. Then, the VSM computes the cosine similarity between the sentences found in each document that also takes $O(nm)$. The total complexity of the VSM is $O(td) + O(nm)$ and both *n* and *m* is less than or equal *t*, which yields $O(td)$ complexity. The complexity analysis of the LSA similarity procedure is complicated. Like the VSM, it depends on the corpus number of terms, but it also depends on the number of topics under which the terms are classified. As we described in section 3.1, the LSA uses the SVD to reduce the dimensions of the documents-terms matrix, and the SVD requires $O(min\{t^2d\},\{td^2\})$ (see [32, 59]).

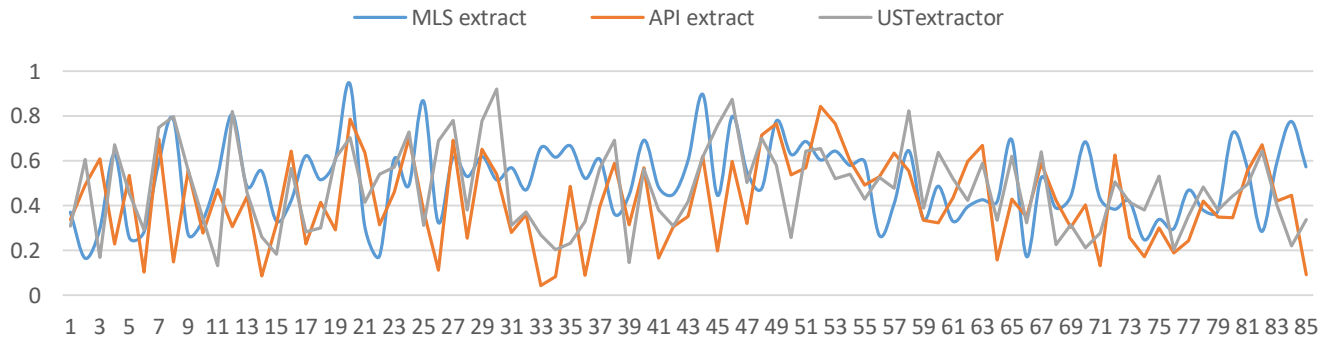


Fig. 11.a Recall Values for the MLS, API, and UTF-8 SUPPORT TOOL Extracts

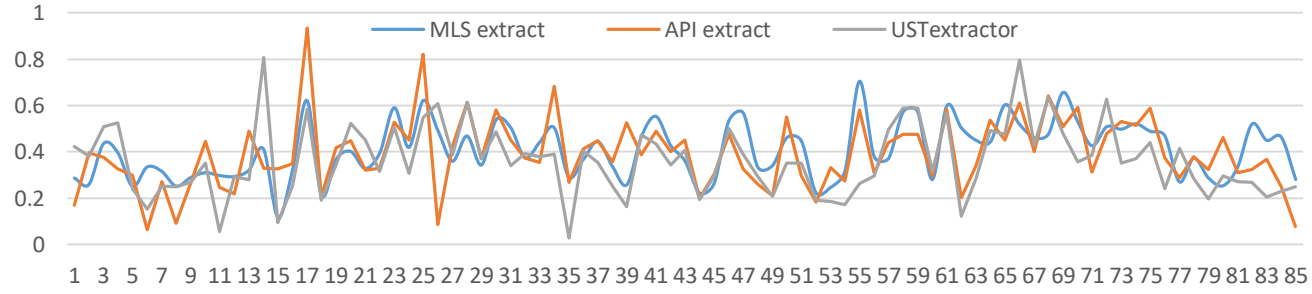


Fig. 11.b Precision Values for the MLS, API, and UTF-8 SUPPORT TOOL Extracts

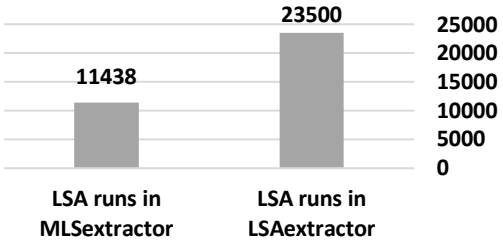


Fig. 12 LSA Number of Runs in LSAExtractor and MLSExtractor for 133 Documents

The LSA provided significant results either in the Containment Evaluation or the Rouge evaluation, and it showed 54% compression rate. The primary challenge faces the use of LSA is the time needed to process the LSA similarity between the sentences. If we have n sentences in a document, this means that we need to run the LSA procedure $n\left(\frac{n-1}{2}\right)$ times. In MLS extraction, we employ the LSA in an efficient way that reduces the number of runs for the LSA function and benefits from the similarity computations that are performed in the Jaccard and VSM similarity procedures, which take less time for processing.

Our results showed that on average we decreased the number of times we run the LSA procedure using the MLS extraction by 52% (see Fig. 12). For example, the LSAExtractor executed the LSA procedure for document number one 435 times whereas the MLS similarity executed the LSA process for the same document 194 times. Fig. 12 shows a comparison between the LSAExtractor and the MLS extractor in the number of runs of the LSA function. The figure reveals that the LSAExtractor executed the LSA procedure 23500 times for 133 documents, whereas the MLSExtractor executed the LSA procedure 11438 times for the same number of documents (52% reduction).

Another critical issue related to the execution time and space is the original matrix, which represents the input to the SVD procedure in the LSA processing. The MLS extraction reduces the dimensionality of this matrix before the execution of the SVD. Fig. 13.a and 13.b present a comparison between the original and the reduced matrix dimensions. Note that Fig. 13.a represents the reduction in the number of terms and Fig. 13.b represents the reduction in the number of sentences. It seems that we should put them in one figure but because we have a large difference in the vertical axis (in one case it represents the terms, and in the second case it represents the sentences) we see to represent them separately. The data was taken from our experiment and for all the processed documents in this experiment, we traced the dimensions for each document inputted to the LSAExtractor ($i\ x\ j\ original\ \ matrix\ see\ definition\ 6$) and the MLSExtractor ($i_{red}\ and\ j_{red}\ definition\ 8\ and\ 9$). In Fig. 13.a and 13.b, the horizontal axis contains the document number, and the documents were sorted according to their size from left to right (the largest on the left). Fig. 13.a and 13.b, show clearly that there is a significant dimensions reduction especially for the large documents that located on the left side of the two figures. As going to the right, the documents sizes become smaller and no important reduction is necessary.

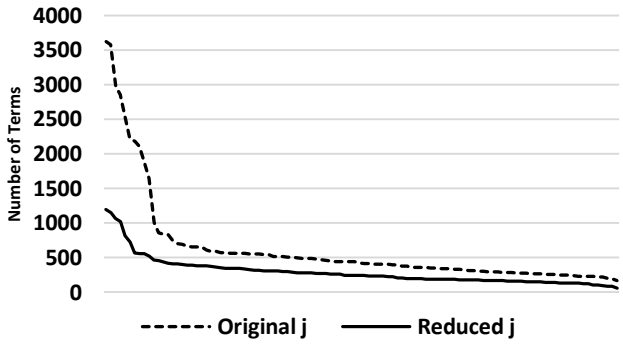


Fig. 13.a the Trend of the Original Number of Processed Terms by the LSAExtractor and the Reduced Number of Processed Terms by MLSExtractor.

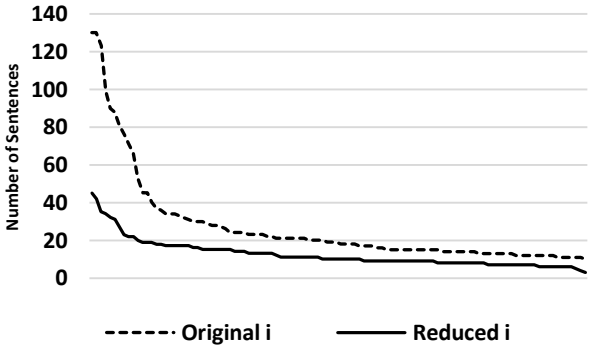


Fig. 13.b the Trend of the Original Number of Processed Sentences by the LSAExtractor and the Reduced Number of Processed Sentences by MLSExtractor.

Table 12 summarises the average of the ratio of reduction for all the documents processed in our experiment. The ratio affected by the fact appeared in Fig. 13, the reduction will be noticeable for the large documents. In the small or moderate size documents the effect of the MLS in reducing the terms-sentences matrix size was limited, which seems logical because the documents already have an acceptable size, and the run of the LSA will not be disturbing, but, for the large

documents the ratio rose to 65%, which means that only 35% of the text needs semantic investigation by the LSA.

Table 12 the Ratio of the Obtained Reduction by MLS

	(Average(j_{red}))	(Average(i_{red}))
Documents with number of sentences > 10	43%	45%
Documents with number of sentences > 40	65%	66%

7 Discussions

We can discuss our work from two perspectives, the extraction model and the new evaluation process. We introduced a new method for text extraction based on the sentences resemblance, and we developed a new evaluation technique that gave us a clear picture of the automatic extracts actual contents. Table 13 presents a sample of the automatically generated extracts.

From our results and evaluation, we said that our technique of extraction has the following features:

No based sentence - such as the user recommendation or the document title- was used as a base of extraction. The extraction process computes the similarity between all the pairs of the sentences found in the document. The similarity value reflects the degree of the verbatim, statistical, or semantic resemblance between any two sentences. For example, the similarity values between the sentence number 4 and the sentence number 11 in document 1 exceeded the threshold value in all the automatic extraction systems that are developed in this work (57%, 99%, 90%, and 57%). And, if we consider the meaning of these two sentences we find that both of them are talking about the appearance of the Beethoven talent in the music from an early age. The sentences are:

قدم أول عمل موسيقي في سن 8 سنوات

He presented the **first musical work** at the **age** of 8 **years**.

ظهر تميزه الموسيقي منذ صغره، فنشرت أولى أعماله وهو في الثانية عشرمن عمره سنه 1783 م.

His musical excellence appeared from a **young age**; he published the **first musical work** when he was twelve-**year-old** in 1783.

Language Independent: No linguistic features were considered during the extraction process, all the similarity calculations were statistical, and they are applicable to another language.

Domain-Independent: Unlike the feature-based extraction and the domain-based extraction proposed by Nekota and McKeown [8], no particular domain features affected our technique of extraction.

Robust deletion process: The deletion procedure used in the extraction systems was robust; it deletes the repeated sentences based on a well-defined process and parameters. It discarded 58% of the text and obtained reasonable levels of HIGH and FULL Containment (27%) (Fig. 3-6). Also from Fig. 8 and 9, the Rouge results showed reasonable recall value (48%), which was higher than the recall values achieved by the researchers in [17, 56] who used the LSA technique in their summarization systems (see Table 1). The MLSExtractor gave 41% F-score values, which were higher than the F-score values obtained in [18, 19, 56]. The precision values of the VSMExtractor surpassed the precision value achieved by Kiyoumarsi in [15] who used the VSM to summarize text documents. Fig. 14 compares the recall value obtained by the MLSExtractor and LSAExtractor with Kiyoumarsi extractor [15], Babar and Patil extractor [17], Chen Extractor [56], Yousefi and Hamey extractor [25], and Tayel Extractor [54]. These extraction systems were developed recently (2015-2017). The authors in [15, 17, 25, 54, 56] used statistical techniques and employ Rouge tool to evaluate their summaries. Fig. 14 clearly shows that the achieved recall values of the MLSExtractor and the LSAExtractor are higher than the other extractors recall values.

Semantic similarity measurement: The use of different approaches for similarity calculation allows us to measure the output of our deletion process at three levels of complexity, simple statistical (Jaccard coefficient), complicated statistical (VSM), and complex statistical with semantic analysis (LSA, MLS). From Fig. 5, 6, 8, and 9, we found that the semantic investigation of the text obtained the most significant Containment values (LOW Containment less than 26%) and Rouge values (recall 60% and precision 41%). Both the MLSExtractor and the LSAExtractor work in the semantic level and both of them obtained convergent Rouge and Containment evaluation results, but the MLS extraction outshone the LSA extraction in the compression rate, and it succeeded to reduce the size of the text to 42% instead of 54% that was achieved by the LSAExtractor. The summary of the MLSExtractor evaluation results can be drawn from Fig. 6, 7, 8, and 9, the CR was 42%, the average recall was 48%, the average precision 40%, with 48% MODERATE Containment, 16% HIGH Containment, and 2% FULL Containment.

Accurate and fair assessment: The Containment evaluation approach provides an accurate judgment about the contents of the automatic extracts. In our evaluation, Rouge provided an indication of the quality of the extract, but it cannot judge accurately the percent of complete sentences shared between the automatic and the reference extracts, and it does not consider the

size of the extract. Only the Containment evaluation showed the superiority of the LSA and the MLS extractions over the Jaccard and the VSM extractions. Fig. 3-7 showed that the MODERATE plus HIGH plus FULL Containment for the JacExtractor was 97%, for the VSMExtractor was 81%, for the LSAExtractor was 74%, and for the MLSExtractor was 66%. If we combine these results with the CR values and implement them in pairs (Containment, CR), we will get the pairs (97%, 79%), (81%, 68%), (74%, 54%), and (66%, 42%) for the JacExtractor, VSMExtractor, LSAExtractor, and MLSExtractor respectively. These pairs are important because they showed the percent of the sentences from the reference extracts found in the automatic extracts to the percent of the text size reduction. For example, in the JacExtractor, 97% of the references extracts sentences found in the automatic extracts at a compression rate of 79% (only 21% from the text removed). These pairs clearly show that the MLSExtractor yielded the most acceptable results by matching the percent of correct retrieved sentences (66%) to the automatic extract size (42% of the original text).

***Performance increase*:** The time complexity analysis that was accomplished in this paper showed that the complexity of the LSA extraction is high comparing with the VSMExtractor and the JacExtractor. The time analysis also showed how the MLS model employed the LSA model in an efficient way. From Fig. 12, the number of runs of the LSA procedure in the MLS extraction is less than the number of runs of the LSA procedure in the classical LSA extraction by 52%. Furthermore, in the MLSExtractor, the SVD processed 35% of the terms and sentences found in the original matrix (especially for large documents, see Fig. 13). This is a significant result because it increases the acceptability of employing the MLS model in the Text Mining fields.

Variable-sized Extracts: The use of a variable size compression rate allows us to create a condensed version of the document that contains all the salient parts of the document and helped us to develop an accurate evaluation. Firstly, only the compression rate was able to show that the MLSExtractor surpassed the other automatic extraction systems. Secondly, if we used fixed CR, we cannot pretend that one extractor- from the four extraction systems developed in this work -performed better than the other extractors because in the fixed-size text extraction the size of the extract prevents the system to include more sentences that may create the difference.

Stable behavior: A comparison with the existing automatic extraction methods was established. The comparison showed the bright side of our extraction method. Besides the significant recall and precision values, the system showed stable behavior, and in most cases, it returns recall and precision values that were close to their mean. The MLS Extractor obtained 0.09 standard deviation for the recall values and 0.14 for the precision values. Comparing that with the existing methods, UTF-8 SUPPORT TOOL and API, we found that our MLS extraction system obtained the most significant standard deviation of the recall and precision values (see Fig. 11.a and 11.b).

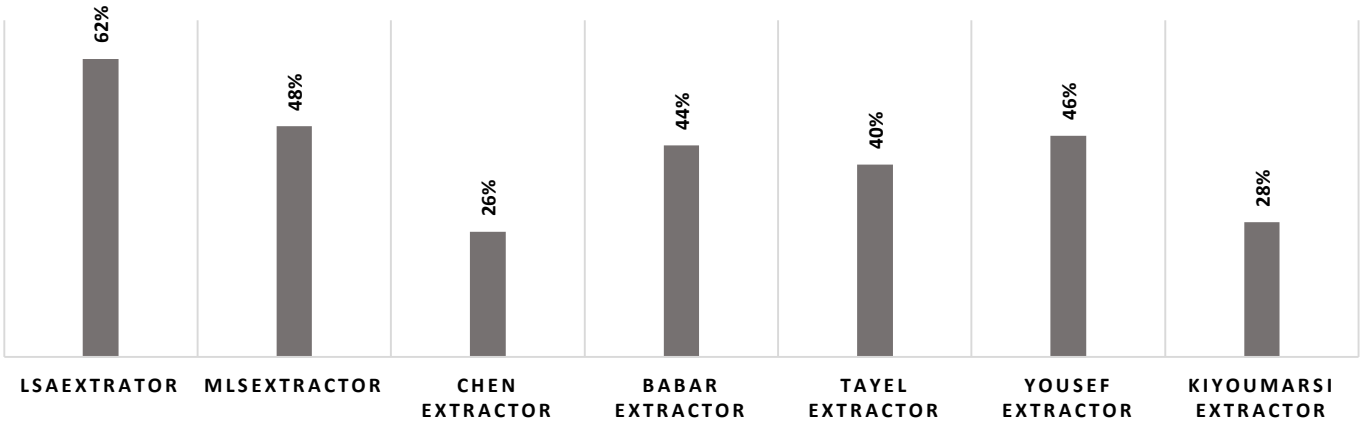


Fig. 14 a Recall Comparison between the MLSExtractor and LSAExtractor with Recent Extractors.

8 Conclusion and Future Work

8.1 Achievements

The paper presents an accurate text extraction method based on the use of an efficient semantic analysis framework. The method uses the centrality feature, and the centrality is computed using a multilayer statistical approach. The multilayer similarity computations are designed to minimize the use of the LSA. To test our method of extraction, we built four entirely separated extraction systems: (1) JacExtractor that is based on the Jaccard coefficient to measure the overlapped terms between two sentences, (2) VSMExtractor that is based on a tf.idf scheme to calculate the cosine similarity, (3) LSAExtractor that is based on the classical use of the LSA, and (4) MLSExtractor that is based on the semantic analysis framework proposed in this work.

Besides the Rouge evaluation, we proposed a new evaluation technique based on the containment of the automatically extracted sentences in the manual extracts relative to the automatic extract size. The achievements of this research can be summarized in the following points:

1. The analysis of the results showed that the proposed extraction method was significant and succeeded to extract a reasonable ratio of the salient parts in the text. Depending on the Containment evaluation, the four extraction methods succeeded to contain a high ratio of the sentences that appeared in the manual extracts. The percent of LOW Containment did not exceed 34% in all cases. Also, among the four extractors implemented in this paper, the LSAExtractor and the MLSExtractor obtained the significant results regarding the extraction quality and the compression rate, and this reflects the importance of investigating the semantic meaning of the text. The JacExtractor and the VSMExtractor obtained high results comparing with the LSAExtractor results in terms of recall, and Containment but they failed to delete large portions of unnecessary text, and this appears clearly by scanning their CR values. On average, the JacExtractor removed only 22 % of the original text and the VSMExtractor removed 35%.
2. Our research showed that the MLS Extraction method remedies the time complexity problem related to the LSA extraction by (1) reducing the number of runs of the LSA similarity procedure and (2) reducing the original matrix dimensions. The MLS extraction method decreased the number of executions of the LSA program by 52% and the original matrix size by 65% and produces roughly the same Rouge and Containment results that were obtained in the classical LSA extraction.
3. The other important conclusions that appeared after comparing our MLS extraction system with two existing extraction systems –UTF 8 SUPPORT tool and API tool - are the stability and accuracy. The MLSExtractor extraction system obtained higher Rouge results than the UTF-8 SUPPORT and API extractors, and the generated extracts by the MLSExtractor had recall and precision values that are very close to their mean. The dispersion ratio was 0.08 for the recall values and 0.14 for the precision values.

Table 13 the Generated Automatic Extracts for Document 2 in Essex Corpus

VSMExtractor Extract	ذكر تقرير إخباري أول من أمس أن شهر مايو المشمس يشهد أكبر عدد من حالات الانتحار. وقال باحثون بريطانيون إن عدد حالات الانتحار يزيد في شهر مايو المشمس ليكون أكثر من أي شهر آخر وهم يعتقدون أن الأمر راجع إلى حالة الطقس. ويقول المسؤول عن الخدمات الصحية في المجموعة البروفيسور كريس تومسون إن هناك علاقة مباشرة بين سطوع الشمس والانتحار. وتبين الإحصائيات أن عدد محاولات الانتحار ارتفع بنسبة 50 % منذ 1990 وأن معظم من أقدموا على الانتحار كانوا من الرجال. News report reported yesterday that the sunny May has the highest number of suicides. British researchers said the number of suicides increased in May to more than any other month and they thought it was due to the weather. Professor Chris Thompson, the group's health services officer, says there is a direct relationship between sun brightness and suicide. Statistics show that the number of suicide attempts has increased by 50% since 1990 and that most of those who committed suicide were men.
JacExtractor Extract	ذكر تقرير إخباري أول من أمس أن شهر مايو المشمس يشهد أكبر عدد من حالات الانتحار. وتقول مجموعة برايوري المتخصصة في بحوث الطب النفسي إن الطقس المشمس الذي عادة ما يساعد الناس في التغلب على كآبتهم يعطيهم كذلك القدرة على اتباع دوافعهم الانتحارية. اوضحت دراسات أخرى أن مستوى السيروتونين يرتفع حسب كمية أشعة الشمس التي يتّعرض لها الشخص. وتبين الإحصائيات أن عدد محاولات الانتحار ارتفع بنسبة 50 % منذ 1990 وأن معظم من أقدموا على الانتحار كانوا من الرجال. News report reported yesterday that the sunny May has the highest number of suicides. Brownie, a specialist in psychiatric research, says sunny weather, which often helps people overcome their depression, also gives them the ability to follow their suicidal motive .Other studies have shown that the level of serotonin increases according to the amount of sunlight the person is exposed to. Statistics show that the number of suicide attempts has increased by 50% since 1990 and that most of those who committed suicide were men.
LSAExtractor extract	LSAExtractor extract ذكر تقرير إخباري أول من أمس أن شهر مايو المشمس يشهد أكبر عدد من حالات الانتحار. وتقول مجموعة برايوري المتخصصة في بحوث الطب النفسي إن الطقس المشمس الذي عادة ما يساعد الناس في التغلب على كآبتهم يعطيهم كذلك القدرة على اتباع دوافعهم الانتحارية. وتبين الإحصائيات أن عدد محاولات الانتحار ارتفع بنسبة 50 % منذ 1990 وأن معظم من أقدموا على الانتحار كانوا من الرجال. News report reported yesterday that the sunny May has the highest number of suicides. Brownie, a specialist in psychiatric research, says sunny weather, which often helps people overcome their depression, also gives them the ability to follow their suicidal motive .Statistics show that the number of suicide attempts has increased by 50% since 1990 and that most of those who committed suicide were men.
MLSExtractor extract	ذكر تقرير إخباري أول من أمس أن شهر مايو المشمس يشهد أكبر عدد من حالات الانتحار. قال باحثون بريطانيون إن عدد حالات الانتحار يزيد في شهر مايو المشمس ليكون أكثر من أي شهر آخر وهم يعتقدون أن الأمر راجع إلى حالة الطقس. ويقول المسؤول عن الخدمات الصحية في المجموعة البروفيسور كريس تومسون إن هناك علاقة مباشرة بين سطوع الشمس والانتحار. وأوضحت دراسات أخرى أن مستوى السيروتونين يرتفع حسب كمية أشعة الشمس التي يتعرض لها الشخص. News report reported yesterday that the sunny May has the highest number of suicides British researchers said the number of suicides increased in May to more than any other month and they thought it was due to the weather. Professor Chris Thompson, the group's health services officer, says there is a direct relationship between sun brightness and suicide. Other studies have shown that the level of serotonin increases according to the amount of sunlight the person is exposed.

8.2 Future Work

In the future, we plan to boost the first and second layer of our multilayer similarity scheme by experimenting more statistical techniques and by embedding an enhanced version of the bushy path and aggregate similarity methods. Also, in this paper, we used an intrinsic evaluation technique to test the quality of our extract, in the future, we want to evaluate our method of extraction under the extrinsic approach and measure the influence of our automatic extraction on the relevancy of the Information Retrieval applications. Thus, our plan involves designing an information retrieval system that uses the output of the MLSExtractor to build a short and informative inverted index.

9 References

1. Rayner K., Elizabeth S R, Michael M E, Mary P C, Rebecca T (2016) So Much to Read, So Little Time How Do We Read, and Can Speed Reading Help? Psychological Science in the Public Interest, 17(1), 4-34.
2. Luhn H P (1957) A Statistical Approach to Mechanize Encoding and Searching of Literary Information. IBM Journal of Research and Development (IBM), 1(4), 309 – 317.
3. Luhn H P (1958) The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 159-165.
4. Kupiec J, Pedersen J, Chen F (1995) A Trainable Document Summarizer. Proceedings of the 18th annual international ACM SIGIR conference on research and development in information retrieval, (pp. 68 – 73). Seattle WA.
5. Mani I (2001) Automatic Summarization.
6. Binwahlan M S, Salim N, Suanmali L (2009) Intelligent Model for Automatic Text Summarization. Information Technology Journal (Asian Network for Scientific Information), 8(8), 1249 - 1255.
7. Mei J-P, Chen L (2012) SumCR: A New Subtopic-Based Extractive Approach for Text Summarization. Knowledge and Information Systems, 31(3), 527 – 545.
8. Nenkova A, McKeown K (2012) A Survey of Text Summarization Techniques. (C. C. Aggarwal, & E. ChengXiang Zhai, Eds.) Mining Text Data, 43-76.
9. Meena Y K, Gopalani D (2015). Domain Independent Framework for Automatic Text Summarization. Procedia Computer Science, 48, 722–727.
10. Gambhir M, Gupta V (2017) Recent automatic text summarization techniques a survey. Artificial Intelligence Review, 47(1), 1-66.
11. Pierre-Etienne G, Guy L (2011) Framework for Abstractive Summarization Using Text-To-Text Generation. MTTG '11 Proceedings of the Workshop on Monolingual Text-To-Text Generation., (pp. 64-73). Strouds.
12. Song S, Huang H, Ruan, T (2018) Abstractive Text Summarization Using LSTM-CNN based deep learning. (S. US, Ed.) Multimed Tools Appl (2018), 1-19.
13. El-Haj M O, Hammo B H (2008) Evaluation of Query-Based Arabic Text Summarization System. 2008 International Conference on Natural Language Processing and Knowledge Engineering, IEEE. , (pp. 1-7). Beijing.
14. Ghwanmeh S, Kanaan G, Al-Shalabi R, Rabab'ah S (2009) Enhanced Algorithm for Extracting the Root of Arabic Words. Computer Graphics, Imaging, and Visualization, 2009. CGIV '09. Sixth International Conference: IEEE, (pp. 388 – 391). Tia.
15. Kiyoumarsi F (2015) Evaluation of Automatic Text Summarizations based on Human Summaries. Procedia - Social and Behavioral Sciences, 192, 83-91.
16. Svore K M, Vanderwende L, Burges C J (2008) Using Signals of Human Interest to Enhance Single-document Summarization. Technical Report, Association for the Advancement of Artificial Intelligence.
17. Babar S, Patil P D (2015) Improving Performance of Text Summarization. Procedia Computer Science, 46, 354-363.
18. Yeh J-Y, Hao-RenKe Yanga W-P, Meng I-H (2005) Text Summarization Using a Trainable Summarizer and Latent Semantic Analysis. Information Processing & Management, 41(1), 75-95.
19. Mashechkin I V, Petrovskiy M I, Popov D S, Tsarev D V (2011) Automatic Text Summarization Using Latent Semantic Analysis. Programming and Computer Software, 37(6), 299–305.
20. Yang R, Bu Z, Xia Z (2012) Automatic Summarization for Chinese Text Using Affinity Propagation Clustering and Latent Semantic Analysis. Web Information Systems and Mining, Lecture Notes in Computer Science.
21. Wang Y, Ma J (2013) A Comprehensive Method for Text Summarization Based on Latent Semantic Analysis. In N. L. Computing. Berlin Heidelb: Springer-Verlag.
22. Ba-Alwi F M, Gaphari G H, Al-Duqaimi F N (2015) Arabic Text Summarization Using Latent Semantic Analysis. British Journal of Applied Science & Technology, 10(2), 1-14.
23. Abdel Fattah M, Ren F (2008) Probabilistic Neural Network Based TextSummarization. International Conference on Natural Language Processing and Knowledge Engineering. Beijing.
24. Abdel Fattah M, Ren F (2009) GA, MR, FFNN, PNN and GMM Based Models for Automatic Text Summarization. Computer Speech & Language, 23(1), 126 - 144.
25. Yousefi A M, Hamey L (2017) Text Summarization Using Unsupervised Deep Learning. Expert Systems with Applications, 68, 93–105.
26. Yanmin C, Bingquan, L, Xiaolong W (2007) Automatic Text Summarization Based on Textual Cohesion. Journal of Electronics, 24(3), 338-346.
27. El-Shishtawy T, El-Ghannam F (2012) Keyphrase Based Arabic Summarizer (KPAS). Informatics and Systems (INFOS), 2012 8th International Conference: IEEE. NLP-7 - NLP-14. Cairo: IEEE.
28. Alruily M, Hammami N, Goudjil M (2013) Using Transitive Construction for Developing Arabic Text Summarization

System. Computer and Information Technology (WCCIT), 2013 World Congress on 2013, 1-2.
29. Ngoc P V, Tran V T (2018) Latent Semantic Analysis using a Dennis Coefficient for, English Sentiment Classification in a Parallel System. International Journal of Computers Communications & Control, 13(3), 408-428.
30. Yates R B, Neto B R (1999) Modern Information Retrieval. Addison-Wesley Longman.
31. Donga T, Haidar A, Tomov S, Dongarra J (2018) Fast SVD for Large-Scale Matrices. Journal of Computational Science, 26, 237–245.
32. He Z, Deng S, Xu X (2006) A Fast Greedy Algorithm for Outlier Mining. Ng WK., Kitsuregawa M., Li J., Chang K. (Eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2006. Lecture Notes in Computer Science, Volume 3918. Springer, 3918.
33. Jing H, Barzilay R, McKeown K, Elhadad M (1998) Summarization Evaluation Methods: Experiments and Analysis. In AAAI Symposium on Intelligent Summarization.
34. Sparck J K, Galliers J R (1995) Evaluating Natural Language Processing Systems: an Analysis and Review.
35. Edmundson, H. P. (1969). New Methods in Automatic Extracting. Journal of the ACM (JACM), 264-285.
36. Mihalcea R, Ceylan H (2007) Explorations in Automatic Book Summarization. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Prague: Association for Computation Linguistics, (pp. 380-389). Prague.
37. Al-Radaideh Q A, Bataineh D Q (2018) A Hybrid Approach for Arabic Text Summarization Using Domain Knowledge and Genetic Algorithms. Cognitive Computation, 10(4), 651–669.
38. Chen Q-C, Wang X-L, Liu B-Q, Wang Y-Y (2002) Subtopic Segmentation of Chinese Document: an Adapted Dotplot Approach. Proceedings of International Conference on Machine Learning and Cybernetics. IEEE, Pages, (pp. 1571 – 1576). Beijing.
39. Harwath D, Hazen T J (2012) Topic Identification Based Extrinsic Evaluation of Summarization Techniques Applied to Conversational Speech. , 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Pages. , (pp. 5073-5076). Kyoto.
40. Halteren H v, Teufel S (2003) Examining the Consensus between Human Summaries: Initial Experiments with Factoid Analysis. , HLT-NAACL-DUC '03 Proceedings of the HLT-NAACL 03 on Text summarization workshop. Nijmegen: Association for Computational Linguistics, (pp. 57-64).
41. Lin C Y (2004). Rouge: A Package for Automatic Evaluation of Summaries. In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004). Barcelona.
42. Lin C Y (2001) SEE. Retrieved from http://www.isi.edu/cyl/SEE.
43. Hassel M (2004) Evaluation of Automatic Text Summarization. Sweden: Licentiate Thesis Stockholm. Retrieved from http://www.csc.kth.se/~xmartin/papers/licthesis_xmartin_notrims.pdf.
44. Ramanujam N, Kaliappan M (2016) An Automatic Multi-document Text Summarization Approach Based on Naïve Bayesian Classifier Using Timestamp Strategy. The Scientific World Journal Volume, 1-10.
45. Froud H, Lachkar A, Ouatik S A (2013) Arabic Text Summarization Based on Latent Semantic Analysis to Enhance Arabic Documents Clustering. International Journal of Data Mining & Knowledge Management Process (IJDKP), 3(1), 79-95.
46. Ferreira R, Cabral L d, Dueire R, Freitasa S F, Cavalcantia G D, Lima R, Favaro L (2013) Assessing Sentence Scoring Techniques for Extractive Text Summarization. Expert System with Applications, 40(14), 5755-5764.
47. Baxendale P (1958) Machine-Made Index for Technical Literature-an Experiment. IBM Journal of Research and Development, 2(4), 354 – 361.
48. Lin C Y (1997) Identify Topics by Concept Signatures. Technical report, Marina Del Rey: Information Sciences Institute.
49. Lin C Y (1995) Topic Identification by Concept Gene. Proceedings of the Thirty-third Conference of the Association of Computational Ling, (pp. 308-310). Boston.
50. Mani I, Bloedorn E, Gates B (1998) Using Cohesion and Coherence Models for Text Summarization. Reston: AAAI Technical Report.
51. Marcu D (1998) Improving Summarization through Rhetorical Parsing Tuning. Workshop on Very Large Corpora. ACL Anthology Network. Pages, (pp. 206-215).
52. Aone C, Okurowski M E, Gorlinsky J (1998) Trainable, Scalable Summarization Using Robust NLP and Machine Learning. The 17th international conference on Computational linguistics: Association for Computational Linguistics, (pp. 62-66). Stroudsburg.
53. Lin C Y (1999) Training a Selection Function for Extraction. Proceedings of the eighth international conference on information and knowledge management. (pp. 55-62). New York.
54. Tayal M A, Raghuwanshi M, Malik L G (2017) ATSSC: Development of an approach based on soft computing for text summarization. Computer Speech and Language, 41, 214–235.
55. Shams R, Hashem M, Hossain A, Akter S R, Gope M (2010) Corpus-based Web Document Summarization using Statistical and Linguistic Approach. Computer and Communication Engineering (ICCCE), 2010 International Conference. IEEE. (pp. 1-6). Kuala Lumpur.
56. Chen K-Y C, Liu S-H, Chen B, Wang H-M, Jan E-E, Hsu W-L, Chen H-H (2015) Extractive broadcast news summarization leveraging recurrent neural network language modeling techniques. IEEE/ACM Transactions on Audio,

Speech and Language Processing (TASLP), 23(8), 1322-1334.
57. Azmia A M, Al-Thanyyan S (2012) A Text Summarizer for Arabic. Computer Speech and Language, 26(4), 260-273.
58. Douzidia F S, Lapalme G (2004) Lakhas, an Arabic Summarization System. Proceedings of DUC'04.
59. Wang Q, Xu J, Craswell N (2013) Regularized Latent Semantic Indexing: A New Approach to Large-Scale Topic Modeling. ACM Trans. Inf. Syst, 31(1), 1-44.
60. Hanandeh E (2013) Building an Automatic Thesaurus to Enhance Information Retrieval. IJCSI International Journal of Computer Science Issues, 10(1).
61. Sankarasubramaniam Y, Ramanathan K, Ghosh S (2014) Text Summarization Using Wikipedia. Information Processing & Management, 50(3), 443–461.
62. Al-Kabia M (2013) Towards improving Khoja rule-based Arabic stemmer. 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), (pp.1–6), IEEE.
63. Al-Kabi M, Kazakzeh A, Abu Ata B, Al-Rababah A, Alsmad I, (2015) A novel root based Arabic stemmer. Journal of King Saud University - Computer and Information Sciences, 27(2), 94-13.
64. Dai S, Diao Q, Zhou C (2005, September) Performance comparison of language models for information retrieval. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 721-730). Springer, Boston, MA.
65. Singh J N, Dwivedi S K (2013) A comparative study on approaches of vector space model in information retrieval. International Journal of Computer Applications, 975, 8887.
66. Luo S, Wang Y, Feng X, Hu Z (2018, September) A Study of Multi-label Event Types Recognition on Chinese Financial Texts. In EuroSymposium on Systems Analysis and Design (pp. 146-158). Springer, Cham.
67. Ghwanmeh S, Kannan G, Al-Shalabi R, Ababneh A (2009). An Enhanced Text-Classification-Based Arabic Information Retrieval System. In Utilizing Information Technology Systems Across Disciplines: Advancements in the Application of Computer Science (pp. 37-44). IGI Global.